fuzzSetOps

January 29, 2021

[1]:
```
import random
import numpy as np
fuzzy_Set_A = dict()
fuzzy_Set_B = dict()
fuzzy_Set_C = dict()
Y = dict()
```

[26]:
```
# fuzzy_Set_A = {"x1": round(random.uniform(0,1),2), "x2": round(random.
↪uniform(0,1),2), "x3": round(random.uniform(0,1),2), "x4": round(random.
↪uniform(0,1),2)}
# fuzzy_Set_B = {"y1": round(random.uniform(0,1),2), "y2": round(random.
↪uniform(0,1),2), "y3": round(random.uniform(0,1),2), "y4": round(random.
↪uniform(0,1),2)}
# fuzzy_Set_C = {"z1": round(random.uniform(0,1),2), "z2": round(random.
↪uniform(0,1),2), "z3": round(random.uniform(0,1),2), "z4": round(random.
↪uniform(0,1),2)}
```

[2]:
```
while True:
    sizeA = int(input("Enter size of Fuzzy Set A"))
    sizeB = int(input("Enter size of Fuzzy Set B"))
    if sizeA != sizeB:
        print("Cant perfom union and intersection of 2 fuzzy sets of different␣
↪lengths")
        continue
    else:
        break

sizeC = int(input("Enter size of Fuzzy Set C"))
```

```
Enter size of Fuzzy Set A 4
Enter size of Fuzzy Set B 4
Enter size of Fuzzy Set C 3
```

[3]:
```
for i in range(sizeA):
    strA = "x"
    strA = strA+"{}".format(i)
    fuzzy_Set_A[strA] = round(random.uniform(0,1),2)
```

```
    strB = "y"
    strB = strB+"{}".format(i)
    fuzzy_Set_B[strB] = round(random.uniform(0,1),2)

for i in range(sizeC):
    strC = "z"
    strC = strC+"{}".format(i)
    fuzzy_Set_C[strC] = round(random.uniform(0,1),2)

fuzzy_Set_A,fuzzy_Set_B,fuzzy_Set_C
```

[3]:
```
({'x0': 0.31, 'x1': 0.45, 'x2': 0.34, 'x3': 0.97},
 {'y0': 0.72, 'y1': 0.49, 'y2': 0.16, 'y3': 0.19},
 {'z0': 0.67, 'z1': 0.88, 'z2': 0.31})
```

[4]:
```
fuzzy_Set_A,fuzzy_Set_B,fuzzy_Set_C
```

[4]:
```
({'x0': 0.31, 'x1': 0.45, 'x2': 0.34, 'x3': 0.97},
 {'y0': 0.72, 'y1': 0.49, 'y2': 0.16, 'y3': 0.19},
 {'z0': 0.67, 'z1': 0.88, 'z2': 0.31})
```

[5]:
```
'''Union'''
Y.clear()
for keyA, keyB in zip(fuzzy_Set_A, fuzzy_Set_B):
    valA = fuzzy_Set_A[keyA]
    valB = fuzzy_Set_B[keyB]
    if valA > valB:
        Y[keyA] = valA
    else:
        Y[keyB] = valB

print('Union of fuzzy sets A and B is :\n', Y)
```

```
Union of fuzzy sets A and B is :
 {'y0': 0.72, 'y1': 0.49, 'x2': 0.34, 'x3': 0.97}
```

[6]:
```
'''Intersection'''
Y.clear()
for keyA, keyB in zip(fuzzy_Set_A, fuzzy_Set_B):
    valA = fuzzy_Set_A[keyA]
    valB = fuzzy_Set_B[keyB]
    if valA < valB:
        Y[keyA] = valA
    else:
        Y[keyB] = valB
```

```
print('Intersection of fuzzy sets A and B is :\n', Y)
```

```
Intersection of fuzzy sets A and B is :
 {'x0': 0.31, 'x1': 0.45, 'y2': 0.16, 'y3': 0.19}
```

[7]:
```
'''Compliment'''
Y.clear()
for keyA in fuzzy_Set_A.keys():
    Y[keyA] = round(1 - fuzzy_Set_A[keyA],2)
print('Compliment of fuzzy set A is :\n', Y)

Y.clear()
for keyB in fuzzy_Set_B.keys():
    Y[keyB] = round(1 - fuzzy_Set_B[keyB],2)
print('Compliment of fuzzy set B is :\n', Y)
```

```
Compliment of fuzzy set A is :
 {'x0': 0.69, 'x1': 0.55, 'x2': 0.66, 'x3': 0.03}
Compliment of fuzzy set B is :
 {'y0': 0.28, 'y1': 0.51, 'y2': 0.84, 'y3': 0.81}
```

[8]:
```
'''Difference'''
Y.clear()
for keyA, keyB in zip(fuzzy_Set_A, fuzzy_Set_B):
    valA = fuzzy_Set_A[keyA]
    valB = fuzzy_Set_B[keyB]
    if valA < round((1 - valB),2):
        Y[keyA] = valA
    else:
        Y[keyB] = round((1 - valB),2)

print('Difference of fuzzy sets A and B is :\n', Y)

Y.clear()
for keyA, keyB in zip(fuzzy_Set_A, fuzzy_Set_B):
    valA = fuzzy_Set_A[keyA]
    valB = fuzzy_Set_B[keyB]
    if valB < round((1 - valA),2):
        Y[keyB] = valB
    else:
        Y[keyA] = round((1 - valA),2)

print('Difference of fuzzy sets B and A is :\n', Y)
```

```
Difference of fuzzy sets A and B is :
 {'y0': 0.28, 'x1': 0.45, 'x2': 0.34, 'y3': 0.81}
Difference of fuzzy sets B and A is :
 {'x0': 0.69, 'y1': 0.49, 'y2': 0.16, 'x3': 0.03}
```

[9]:
```
'''Cartesion Product'''
AXB = np.zeros((sizeA,sizeB))
BXC = np.zeros((sizeB,sizeC))
i=0
for valA in fuzzy_Set_A.values():
    j = 0
    for valB in fuzzy_Set_B.values():
        if valA < valB:
            AXB[i][j] = valA
        else:
            AXB[i][j] = valB
        j = j + 1
    i = i + 1
R = AXB
print('Relation R ( A x B) is : \n', R)

i = 0
for valB in fuzzy_Set_B.values():
    j = 0
    for valC in fuzzy_Set_C.values():
        if valB < valC:
            BXC[i][j] = valB
        else:
            BXC[i][j] = valC
        j = j + 1
    i = i + 1
S = BXC
print('Relation S ( B x C) is : \n', S)
```

```
Relation R ( A x B) is :
 [[0.31 0.31 0.16 0.19]
 [0.45 0.45 0.16 0.19]
 [0.34 0.34 0.16 0.19]
 [0.72 0.49 0.16 0.19]]
Relation S ( B x C) is :
 [[0.67 0.72 0.31]
 [0.49 0.49 0.31]
 [0.16 0.16 0.16]
 [0.19 0.19 0.19]]
```

[10]:
```
'''max-min composition'''
ROS = np.zeros((sizeA,sizeC))
for i in range(0,sizeA):
    for j in range(0,sizeC):
        element = []
        for k in range(0,sizeB):
            if R[i][k] < S[k][j]:
```

```
                element.append(R[i][k])
            else:
                element.append(S[k][j])
        print(element)
        print()
        ROS[i][j] = max(element)

print('Max-min composition ROS is :\n', ROS)
```

```
[0.31, 0.31, 0.16, 0.19]

[0.31, 0.31, 0.16, 0.19]

[0.31, 0.31, 0.16, 0.19]

[0.45, 0.45, 0.16, 0.19]

[0.45, 0.45, 0.16, 0.19]

[0.31, 0.31, 0.16, 0.19]

[0.34, 0.34, 0.16, 0.19]

[0.34, 0.34, 0.16, 0.19]

[0.31, 0.31, 0.16, 0.19]

[0.67, 0.49, 0.16, 0.19]

[0.72, 0.49, 0.16, 0.19]

[0.31, 0.31, 0.16, 0.19]

Max-min composition ROS is :
 [[0.31 0.31 0.31]
 [0.45 0.45 0.31]
 [0.34 0.34 0.31]
 [0.67 0.72 0.31]]
```

[ ]:

[ ]: