w0vjv5gnl

December 6, 2024

0.1 Ramesh Chandra Soren

Enrollment No: 2022CSB086
Department: Computer Science and Technology

1 Fuzzy Set Temperature Analysis

1.1 Question 2

Let C(x) and D(x) be two fuzzy sets of temperature (Celsius) in a city, defined by the following membership functions:

1.1.1 a. Fuzzy set C: "Cold"

```
C(x) = { 0,              if x <= -10 (extremely cold)
         (x+10)/10,      if -10 < x < 0 (very cold)
         1,              if 0 <= x <= 5 (cold)
         0,              if x > 5 (not cold)
       }
```

1.1.2 b. Fuzzy set D: "Warm"

```
D(x) ={ 0,               if x < 25 (Not warm)
        (x-25)/10,       if 25 <= x < 35 (moderately warm)
        1,               if 35 <= x <= 40 (warm)
        0,               if x > 40 (extremely warm)
      }
```

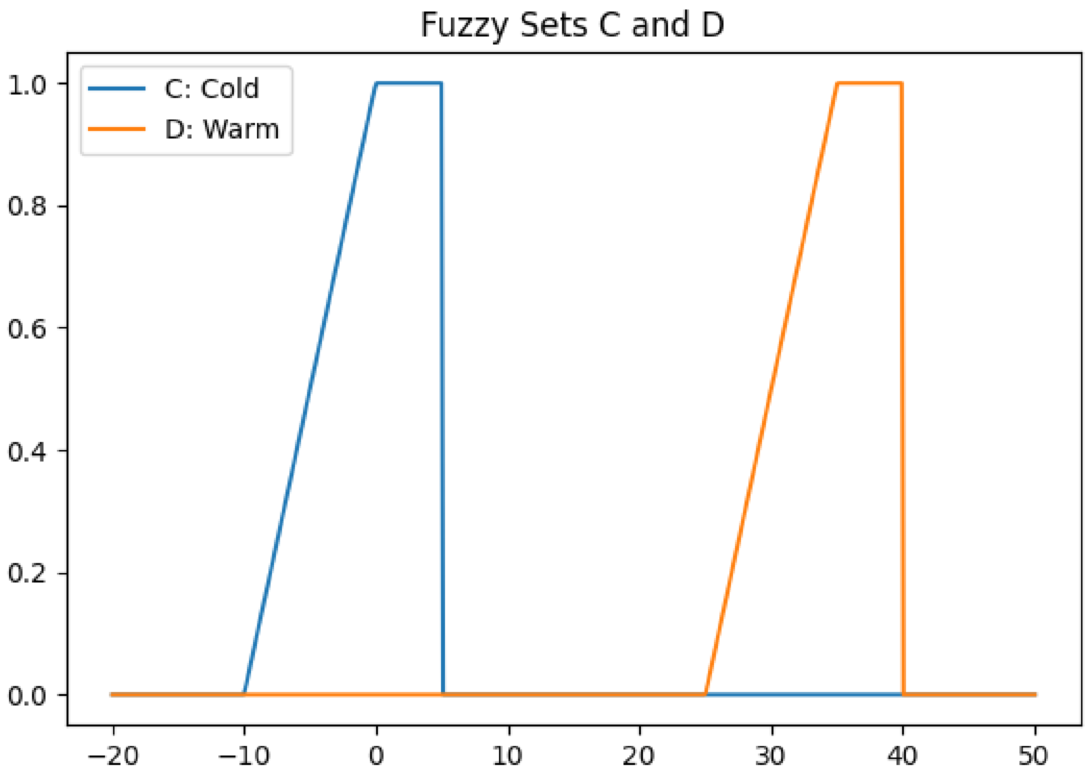
1.2 Tasks

Perform the following operations to obtain the result:

1. Max-Min Composition: (C D)x = max(xi X min(C(xi), D(xi)))
2. Max-Product Composition: (C D)x = max(xi X (C(xi) * D(xi)))

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt

def mu_C(x):
    if x < -10:  # Indent the code block within the function
```

```python
        return 0
    elif -10 < x < 0:
        return (x + 10) / 10
    elif 0 < x < 5:
        return 1
    else:
        return 0

def mu_D(x):
    if x < 25:   # Indent the code block within the function
        return 0
    elif 25 < x < 35:
        return (x - 25) / 10
    elif 35 < x < 40:
        return 1
    else:
        return 0

# You'll need to define x and calculate y_C, y_D based on your fuzzy logic␣
↪before plotting
# For example:
x = np.linspace(-20, 50, 500)  # Define a range of x values
y_C = [mu_C(i) for i in x]  # Calculate membership values for C
y_D = [mu_D(i) for i in x]  # Calculate membership values for D

# Plotting
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
plt.plot(x, y_C, label='C: Cold')
plt.plot(x, y_D, label='D: Warm')
plt.title('Fuzzy Sets C and D')
plt.legend()
plt.show()
```



[ ]:
```python
def mu_C(x):
    """
    Membership function for fuzzy set C: "Cold"
    """
    if x < -10:
        return 0
    elif -10 <= x < 0:
        return (x + 10) / 10  # Linear increase from 0 to 1
    elif 0 <= x < 5:
        return 1
    else:
        return 0

# Example usage:
temp = -5
membership_cold = mu_C(temp)
print(f"Membership of {temp} degrees in 'Cold' set: {membership_cold}")
```

```
Membership of -5 degrees in 'Cold' set: 0.5
```

[ ]:
```python
def mu_D(x):
    """
```

```python
    Membership function for fuzzy set D: "Warm"
    """
    if x < 25:
        return 1
    elif 25 <= x < 35:
        return (x - 25) / 10  # Linear increase from 0 to 1
    elif 35 <= x < 40:
        return 1
    else:
        return 0

# Example usage:
temp = 30
membership_warm = mu_D(temp)
print(f"Membership of {temp} degrees in 'Warm' set: {membership_warm}")
```

```
Membership of 30 degrees in 'Warm' set: 0.5
```

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

# Define temperature ranges for Cold and Warm sets
temp_C = np.linspace(-15, 10, 100)  # Range for "Cold" set
temp_D = np.linspace(20, 45, 100)  # Range for "Warm" set

# Calculate Fuzzy Relation R (Max-Min Composition)
R_max_min = np.zeros((len(temp_C), len(temp_D)))
for i, c in enumerate(temp_C):
    for j, d in enumerate(temp_D):
        R_max_min[i, j] = min(mu_C(c), mu_D(d))

# Calculate Fuzzy Relation R (Max-Product Composition)
R_max_product = np.zeros((len(temp_C), len(temp_D)))
for i, c in enumerate(temp_C):
    for j, d in enumerate(temp_D):
        R_max_product[i, j] = mu_C(c) * mu_D(d)

print("Fuzzy Relation R (Max-Min):\n", R_max_min)
print("\nFuzzy Relation R (Max-Product):\n", R_max_product)
```

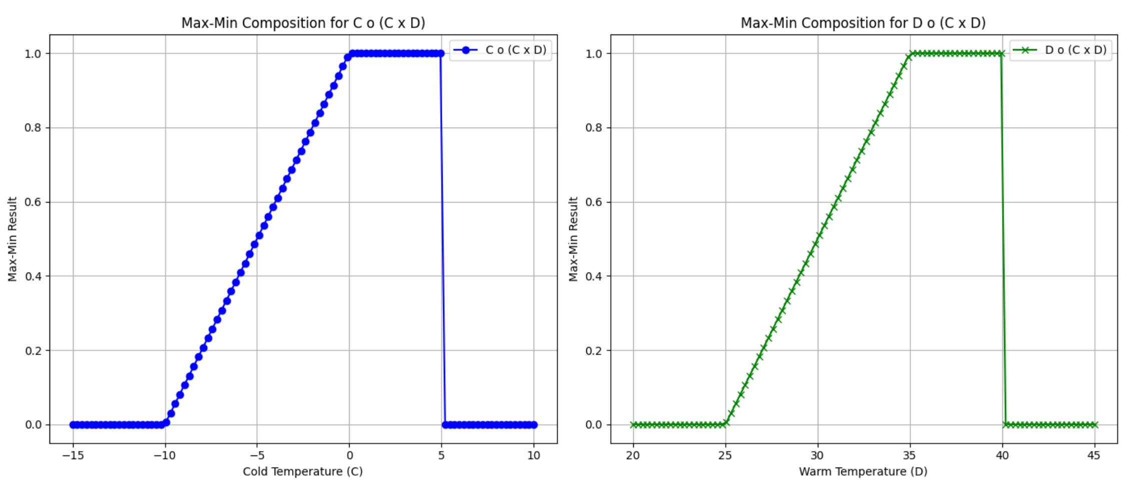
```
Fuzzy Relation R (Max-Min):
 [[0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 …
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]]
```

```
Fuzzy Relation R (Max-Product):
 [[0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 …
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]]
```

[ ]:

```python
def mu_C(x):
    if x <= -10:
        return 0
    elif -10 < x <= 0:
        return (x + 10) / 10
    elif 0 < x <= 5:
        return 1
    else:
        return 0

def mu_D(x):
    if x <= 25:
        return 0
    elif 25 < x <= 35:
        return (x - 25) / 10
    elif 35 < x <= 40:
        return 1
    else:
        return 0

# Define temperature ranges for Cold and Warm sets
temp_C = np.linspace(-15, 10, 100)  # Range for "Cold" set
temp_D = np.linspace(20, 45, 100)  # Range for "Warm" set

# Composition C o (C x D) using Max-Min
result_C = []
for i in range(len(temp_C)):
    max_value = 0
    for j in range(len(temp_D)):
        max_value = max(max_value, min(mu_C(temp_C[i]), R_max_min[i][j]))
    result_C.append(max_value)

# Composition D o (C x D) using Max-Min
result_D = []
for j in range(len(temp_D)):
    max_value = 0
    for i in range(len(temp_C)):
```

```
        max_value = max(max_value, min(mu_D(temp_D[j]), R_max_min[i][j]))
    result_D.append(max_value)

# Plotting the results
plt.figure(figsize=(14, 6))

# Plot result_C (Composition C o (C x D))
plt.subplot(1, 2, 1)
plt.plot(temp_C, result_C, label='C o (C x D)', color='blue', marker='o')
plt.title('Max-Min Composition for C o (C x D)')
plt.xlabel('Cold Temperature (C)')
plt.ylabel('Max-Min Result')
plt.grid(True)
plt.legend()

# Plot result_D (Composition D o (C x D))
plt.subplot(1, 2, 2)
plt.plot(temp_D, result_D, label='D o (C x D)', color='green', marker='x')
plt.title('Max-Min Composition for D o (C x D)')
plt.xlabel('Warm Temperature (D)')
plt.ylabel('Max-Min Result')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```

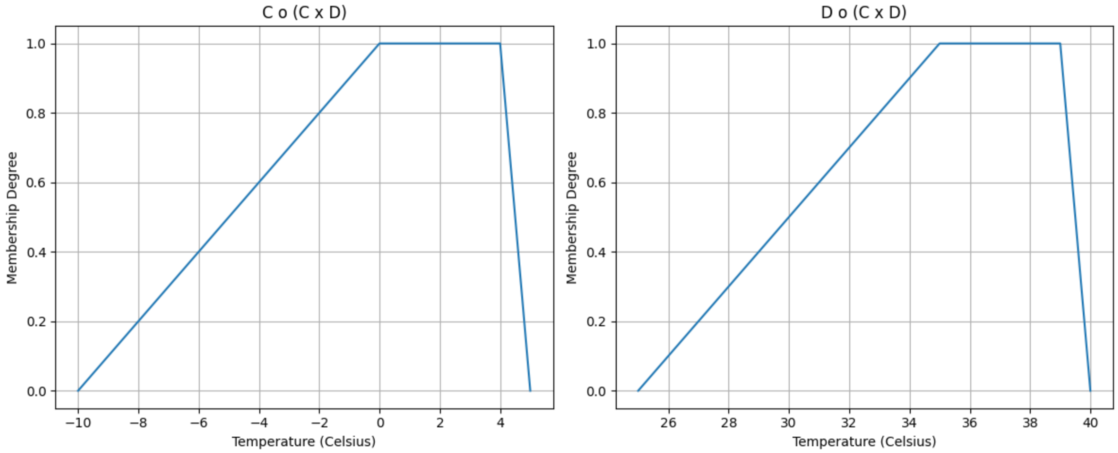


[ ]:

```
# Composition C o (C x D) using Max-Product
result_C = []
for i in range(len(temp_C)):
    max_value = 0
```

```python
    for j in range(len(temp_D)):
        max_value = max(max_value, min(mu_C(temp_C[i]), R_max_min[i][j]))
    result_C.append(max_value)

# Composition D o (C x D) using Max-Product
result_D = []
for j in range(len(temp_D)):
    max_value = 0
    for i in range(len(temp_C)):
        max_value = max(max_value, min(mu_D(temp_D[j]), R_max_min[i][j]))
    result_D.append(max_value)
```

[ ]:
```python
# Plot the results
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(temp_C, result_C)
plt.xlabel('Temperature (Celsius)')
plt.ylabel('Membership Degree')
plt.title('C o (C x D)')
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(temp_D, result_D)
plt.xlabel('Temperature (Celsius)')
plt.ylabel('Membership Degree')
plt.title('D o (C x D)')
plt.grid(True)

plt.tight_layout()
plt.show()

# Print the results
print("Result of C o (C x D):", result_C)
print("Result of D o (C x D):", result_D)
```




```
Result of C o (C x D): [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1, 1,
1, 1, 0]
Result of D o (C x D): [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1, 1,
1, 1, 0]
```