Нечеткое множество

Презентацию подготовили:

Студенты группы 5130901/10202:

Жилкина Лада
Сергиенко Кирилл
Лодочникова Владлена
Чаплин Виталий

Студенты группы 5130901/10201:

Стеблецов Роман
Вилисова Дарья

Определение

$$A = \mu_A x / x$$

Нечеткое множество (fuzzyset) представляет собой совокупность элементов произвольной природы, относительно которых нельзя точно утверждать - обладают ли эти элементы некоторым характеристическим свойством, которое используется для задания нечеткого множества.

Пример

Пусть универсальное множество X соответствует множеству возможных значений толщин изделия от 10мм до 40мм с дискретным шагом 1мм. Нечеткое множество A, соответствующее нечеткому понятию «малая толщина изделия», может быть представлено в следующем виде:

$$A=1/10+0{,}9/11+0{,}8/12+0{,}7/13+0{,}5/14+0{,}3/15+0{,}1/16+0/17+\ \ldots\ +0/40$$

Носителем нечеткого множества A будет конечное подмножество (дискретный носитель):

$$S_A=10;11;12;13;14;15;16.$$

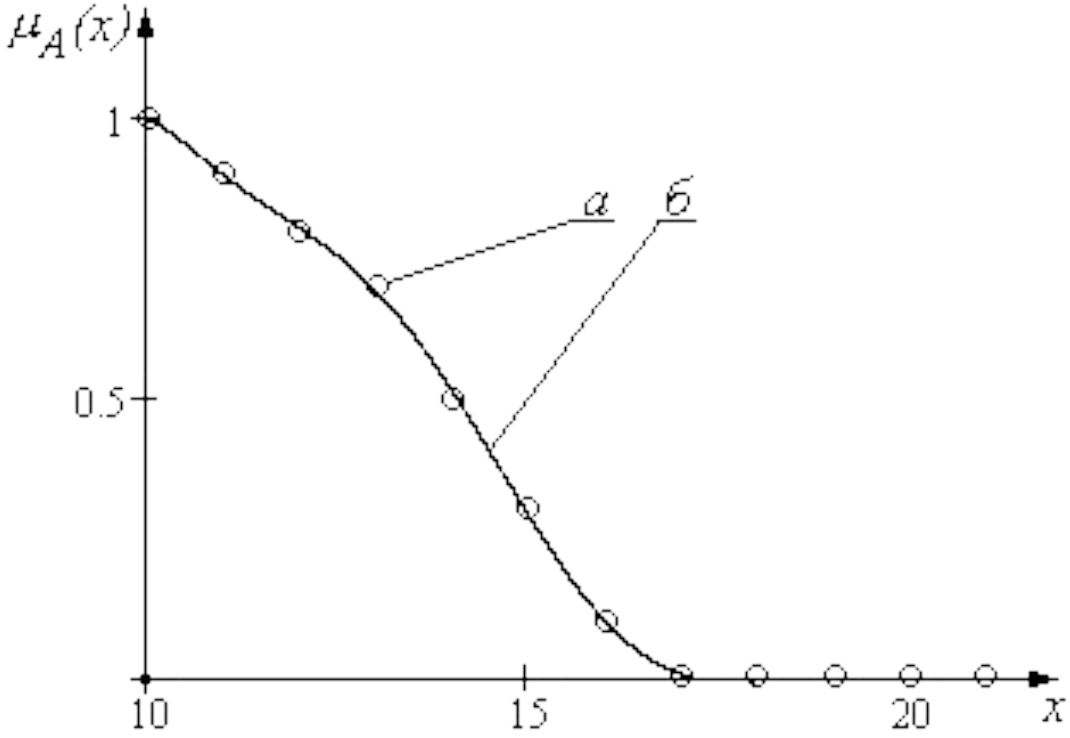
Рассмотрим другой случай:

Универсальное множество X является множеством действительных чисел от 10 до 40, т.е. толщина изделия может принимать все возможные значения в этих пределах.

В таком случае носителем нечеткого множества A является отрезок

$$S_A=[10,16]$$

Представление в виде графика



Класс FuzzySet

```
class FuzzySet:
    # создание
    def __init__(self, elements):
        self._set = elements
        self.fuzzy_sort()
```

Методы для работы с множеством

- получение списка элементов множества
- получение значение принадлежности элемента
- добавление элементов
- изменение степени принадлежности элемента
- проверка на корректность данных

Получение списка элементов множества

```
# множество
def get_set(self):
    return self._set
```

```
A:  [[1, 1], [2, 0.8], [3, 0.6], [4, 0.3], [5, 0.1]]
B:   [[6, 0.9], [7, 0.5], [8, 0.2]]
```

Получение значения принадлежности элемента

```python
# значение принадлежности элемента
def get_membership_value(self, element):
    elements = self.get_set()
    s = [e[0] for e in elements]
    if element in s:
        return self.get_set()[s.index(element)][1]
    else:
        print("Element not found")
        return 0
```

```python
A = FuzzySet([[1, 1], [2, 0.8], [3, 0.6], [4, 0.3], [5, 0.1]])
B = FuzzySet([[1, .9], [2, .9], [3, 0.2]])
print("A: ", A.get_membership_value(4))
print("B:  ", B.get_membership_value(4))
```

```
A:  0.3
Element not found
B:   0
```

Добавление элементов

```
# добавление элемента(ов)
def add_elements(self, elements):
    for element in elements:
        self._set.append(element)
    self.fuzzy_sort()
```

```
A = FuzzySet([[1, 1], [2, 0.8], [3, 0.6], [4, 0.3], [5, 0.1]])
B = FuzzySet([[1, .9], [2, .9], [3, 0.2]])
elements = [[10, 1], [11, 0.2], [12, 0]]
print("A: ", A.get_set())
print("B:  ", B.get_set())
print("Elements:  ", elements)
A.add_elements(elements)
B.add_elements(elements)
print("A: ", A.get_set())
print("B:  ", B.get_set())
```

```
A:  [[1, 1], [2, 0.8], [3, 0.6], [4, 0.3], [5, 0.1]]
B:   [[1, 0.9], [2, 0.9], [3, 0.2]]
Elements:   [[10, 1], [11, 0.2], [12, 0]]
A:  [[1, 1], [2, 0.8], [3, 0.6], [4, 0.3], [5, 0.1], [10, 1], [11, 0.2], [12, 0]]
B:   [[1, 0.9], [2, 0.9], [3, 0.2], [10, 1], [11, 0.2], [12, 0]]
```

Изменение степени принадлежности элемента

```python
# изменение степени принадлежности элемента
def set_membership_value(self, element, value):
    if element in self.get_elements():
        self._set[self.get_elements().index(element)][1] = value
        self.fuzzy_sort()
    else:
        print("Element not found")
```

```python
A = FuzzySet([[1, 1], [2, 0.8], [3, 0.6]])
print("A: ", A.get_set())
A.set_membership_value(1, 0)
A.set_membership_value(2, 1)
A.set_membership_value(3, 0.1)
print("A: ", A.get_set())
```

```
A:  [[1, 1], [2, 0.8], [3, 0.6]]
A:  [[1, 0], [2, 1], [3, 0.1]]
```

Коррекция данных

```python
# проверка на корректность данных
def fuzzy_sort(self):
    sorted_set = []
    seen = set()
    for element in self.get_set():
        if not isinstance(element[1], (int, float)):
            print("Error for element ", element, ": membership value is not a number. Aborting element\n")
        elif element[0] in seen:
            print("Warning for element ", element, ": duplicate encountered. Aborting element\n")
        elif element[1] > 1:
            print("Warning for element ", element, ": membership value of higher than 1 encountered. Setting "
                                                    "membership value to 1\n")
            element[1] = 1
            sorted_set.append(element)
            seen.add(element[0])
        elif element[1] < 0:
            print("Warning for element ", element, ": membership value of lower than 0 encountered. Setting "
                                                    "membership value to 0\n")
            element[1] = 0
            sorted_set.append(element)
            seen.add(element[0])
        elif element[0] not in seen:
            sorted_set.append(element)
            seen.add(element[0])
    self._set = sorted_set
```

```python
A = FuzzySet([[1, 'c']])
B = FuzzySet([[1, 1], [1, 0.2]])
C = FuzzySet([[1, 2]])
D = FuzzySet([[1, -2]])
```

```
Error for element  [1, 'c'] : membership value is not a number. Aborting element

Warning for element  [1, 0.2] : duplicate encountered. Aborting element

Warning for element  [1, 2] : membership value of higher than 1 encountered. Setting membership value to 1

Warning for element  [1, -2] : membership value of lower than 0 encountered. Setting membership value to 0
```

Свойства

- носитель
- пустота
- высота
- субнормальность
- точечность
- унимодальность
- точка перехода
- ядро

Носитель

нечеткого множества - подмножество A множества X, содержащее те элементы из X, для которых значения функции принадлежности больше нуля.

```
# вывод носителя нечёткого множества
def support(self):
    return [elem[0] for elem in self.get_set() if self.get_membership_value(elem[0]) > 0]
```

тесты:

```
A:  [[30, 0.2], [104, 0.0]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Support: [30]
Support: [30, 10, 9, 8, 12]
```

Пустота

Нечеткое множество называется пустым, если $\forall\ x \in X, \mu x = 0$

```python
# пустота
def is_empty(self):
    for element in self.get_set():
        if element[1] != 0:
            return False
    return True
```

```
A:  [[1, 1], [False, 1], [3, 0], ['a', 0.8], ['2', 0.3]]
B:   [[1, 0.1], ['2', 0.8], [3, 0.4], [9, 0.7]]
C:   [[1, 0.3], [False, 1], [3, 0], ['a', 0.8], ['2', 0.1]]
D:   [[1, 0], [False, 0], [3, 0], ['a', 0], ['2', 0]]
I:   []
empty A:  False
empty B:  False
empty C:  False
empty D:  True
empty I:  True
```

Высота

Высотой нечеткого множества A называется верхняя граница его функции принадлежности.

```python
# высота
def get_height(self):
    max_membership_value = max(self.get_set(), key=lambda x: x[1])[1]
    return max_membership_value
```

```
A:  [[30, 0.2], [6, 0.9], [3333, 1e-07], [9, 0.3]]
B:  [[30, 0.8], [7, 0.6], [44, 0.68], [9, 0.99]]
Height A:  0.9
Height B:   0.99
```

Субнормальность

Если sup μx < 1, нечеткое множество называется субнормальным.

```
# проверка множества на субнормальность
def is_subnormal(self):
    for elem in self.get_set():
        if self.get_membership_value(elem[0]) == 1:
            return False
    return True
```

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Is subnormal: True
Is subnormal: False
```

Точечность

Нечеткое множество называется точечным, если μx > 0 только для 1 точки x универсального множества X.

```python
# точечность
def is_point_set(self):
    set_ = self.get_set()
    count_ = 0

    for pair in set_:
        key, value = pair
        if value != 0:
            count_ += 1

    return count_ == 1
```

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
C:   [[30, 0.8], [5, 0.0], [7, 0.0], [19, 0.0]]
Is point set: True
Is point set: False
Is point set: True
```

Унимодальность

Нечеткое множество называется унимодальным, если μx = 1 только для одной точки x (моды) универсального множества X.

```python
# унимодальность
def is_unimodal(self):
    set_ = self.get_set()
    count_ones = 0

    for pair in set_:
        key, value = pair
        if value == 1:
            count_ones += 1

    return count_ones == 1
```

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
C:   [[30, 0.8], [5, 1], [7, 0.0], [19, 0.0]]
Is unimodal: False
Is unimodal: False
Is unimodal: True
```

Ядро

Ядром нечеткого множества A, определенного на универсальном множестве X, называется четкое множество coreA, элементы которого удовлетворяют условию coreA=x∈X | μx = 1

```python
# ядро
def find_core(self):
    set_ = self.get_set()
    core = []

    for pair in set_:
        key, value = pair[0], pair[1]
        if value == 1:
            core.append(key)

    return core
```

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
C:   [[30, 0.8], [5, 1], [7, 0.0], [19, 0.0]]
Core: []
Core: [10, 9]
Core: [5]
```

Нечеткий срез

Альфа-сечением (или α-срезом) нечеткого
множества Ã называется четкое подмножество универсального
множества U, элементы которого имеют степени принадлежности
большие или равные α.

```
# реализация нечёткого среза
def fuzzy_slice(self, a):
    return [elem[0] for elem in self.get_set() if self.get_membership_value(elem[0]) >= a]
```

Тестирование

```
print("Fuzzy slice:", A.fuzzy_slice(1))
print("Fuzzy slice:", A.fuzzy_slice(0.5))
print("Fuzzy slice:", A.fuzzy_slice(0.2))
print("Fuzzy slice:", B.fuzzy_slice(1))
print("Fuzzy slice:", B.fuzzy_slice(0.7))
```

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Fuzzy slice: []
Fuzzy slice: []
Fuzzy slice: [30]
Fuzzy slice: [10, 9]
Fuzzy slice: [30, 10, 9, 8]
```

Точки перехода

Точка перехода нечеткого множества A - такой элемент $x \in U$, для которого $\mu_A(x) = 0,5$

```python
# вывод точек перехода
def transition_point(self):
    return [elem[0] for elem in self.get_set() if self.get_membership_value(elem[0]) == 0.5]

# проверка точек перехода
def is_transition_point(self, point):
    for elem in self.get_set():
        if self.get_membership_value(elem[0]) == 0.5 and elem[0] == point:
            return True
    return False
```

Тестирование

```
print("List transition points: ", A.transition_point())
print("List transition points: ", B.transition_point())
print("Is transition point: ", A.is_transition_point(5))
print("Is transition point: ", B.is_transition_point(17))
print("Is transition point: ", B.is_transition_point(10))
print("Is transition point: ", B.is_transition_point(12))
```

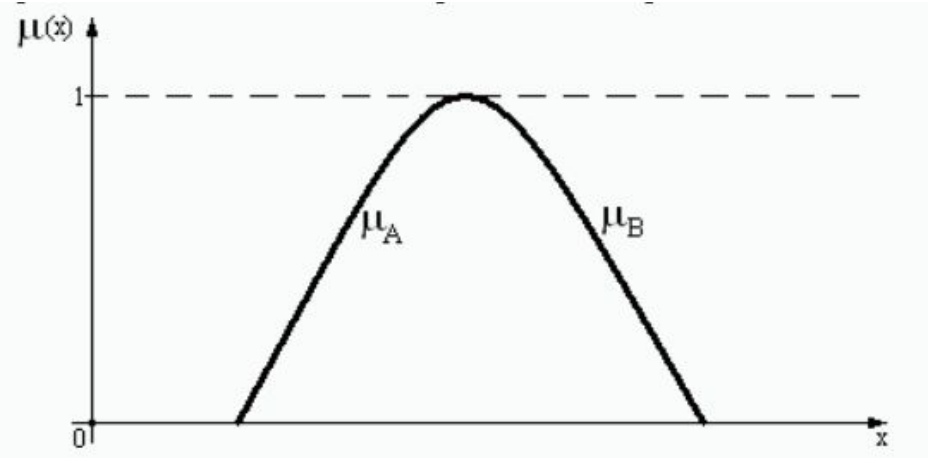
```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
List transition points:  []
List transition points:  [12]
Is transition point:  False
Is transition point:  False
Is transition point:  False
Is transition point:  True
```

Реализованные операции с множеством

- равенство
- объединение
- пересечение
- дополнение
- разность
- симметрическая разность
- дизъюнктивная сумма

Равенство

A и B - нечеткие множества на универсальном множестве X. A и B равны, если $\forall x \in X \mu_A x = \mu_B x.$



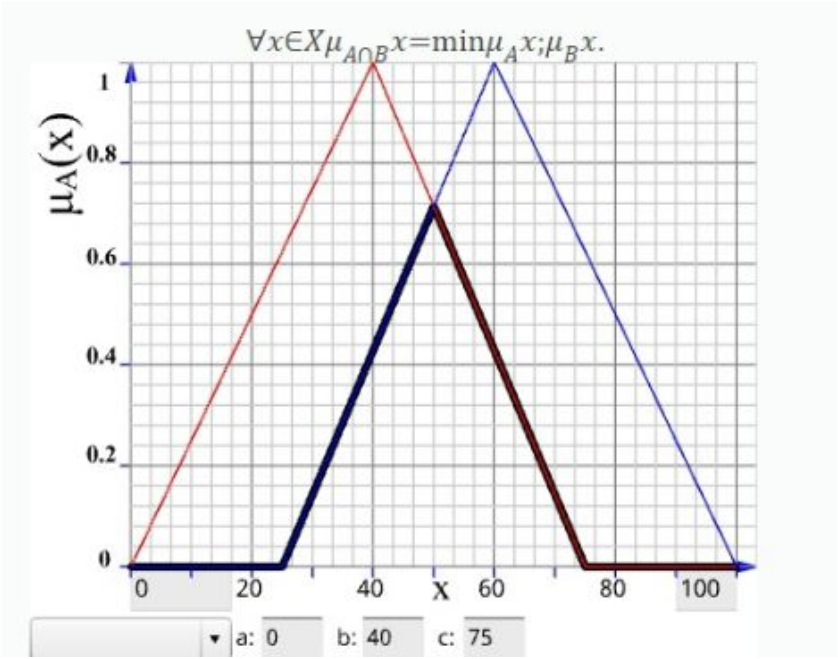
Равенство

$$\forall x \in X \mu_A x = \mu_B x.$$

```
# равенство
def equal(set_a, set_b):
    for element in set_a.get_set():
        if element[0] not in set_b.get_elements():
            return False
        else:
            if element[1] != set_b.get_membership_value(element[0]):
                return False

    for element in set_b.get_set():
        if element[0] not in set_a.get_elements():
            return False
        else:
            if element[1] != set_a.get_membership_value(element[0]):
                return False
    return True
```

Пересечение



Пересечение нечетких множеств A и B, заданных на универсальном множестве X, — это наибольшее нечеткое множество A∩B, содержащиеся одновременно и в A, и в B с функцией принадлежности, заданной следующим образом:

$$\forall x \in X \mu_{A \cap B} x = \min \mu_A x; \mu_B x.$$

Пересечение

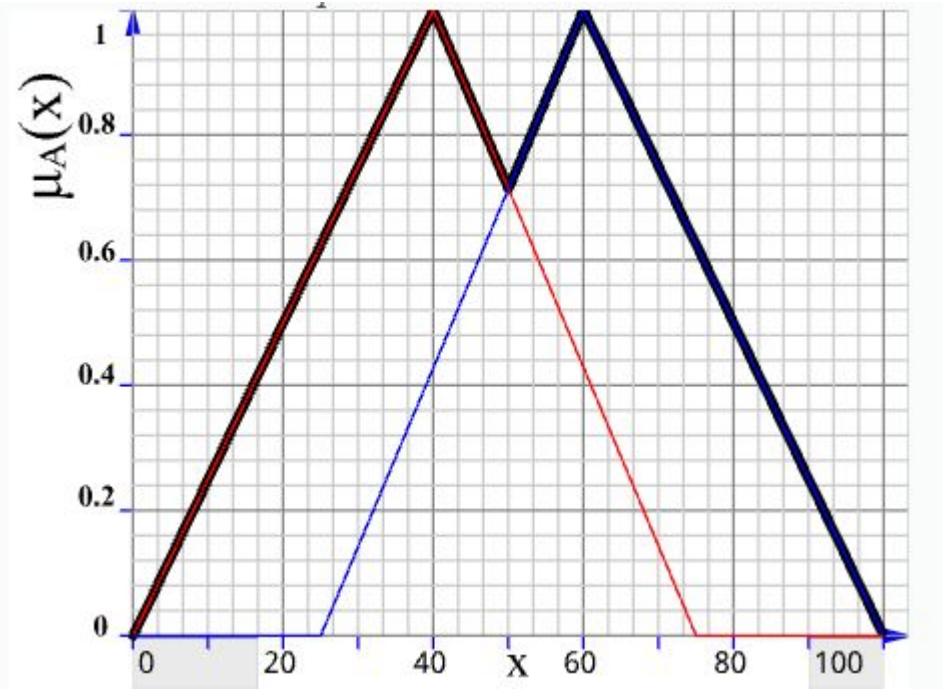
$$\forall x \in X \mu_{A \cap B} x = \min \mu_A x; \mu_B x.$$

```
# пересечение
def intersection(set_a, set_b):
    intersection_set = []
    elements_b = set_b.get_elements()
    for element in set_a.get_set():
        if element[0] in elements_b:
            if element[1] < set_b.get_set()[elements_b.index(element[0])][1]:
                intersection_set.append(element)
            else:
                intersection_set.append(set_b.get_set()[elements_b.index(element[0])])
    return FuzzySet(intersection_set)
```

Объединение

Объединение - наименьшее нечеткое множество $A \cup B$, включающее как A, так и B, с функцией принадлежности

$$\forall x \in X \mu_{A \cup B} x = \max \mu_A x; \mu_B x.$$



Объединение

$$\forall x \in X \mu_{A \cup B} x = \max \mu_A x; \mu_B x.$$

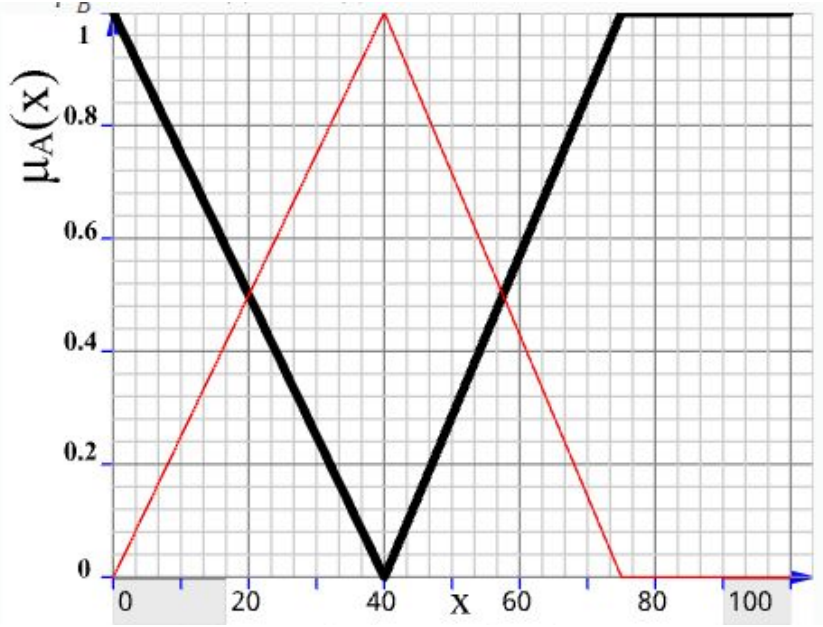
```python
# объединение
def union(set_a, set_b):
    union_set = []
    for element in set_a.get_set():
        if element[0] in set_b.get_elements():
            if element[1] > set_b.get_membership_value(element[0]):
                union_set.append(element)
            else:
                union_set.append([element[0], set_b.get_membership_value(element[0])])
        else:
            union_set.append(element)

    for element in set_b.get_set():
        if element[0] not in [elem[0] for elem in union_set]:
            if element[0] in set_a.get_elements():
                if element[1] > set_a.get_membership_value(element[0]):
                    union_set.append(element)
                else:
                    union_set.append([element[0], set_a.get_membership_value(element[0])])
            else:
                union_set.append(element)

    return FuzzySet(union_set)
```

Дополнение

A и B – нечеткие множества с множеством принадлежностей характеристических функций M=0;1, заданные на универсальном множестве X. Говорят, что A и B дополняют друг друга, если $\forall x \in X \mu_A x = 1 - \mu_B x.$



Дополнение

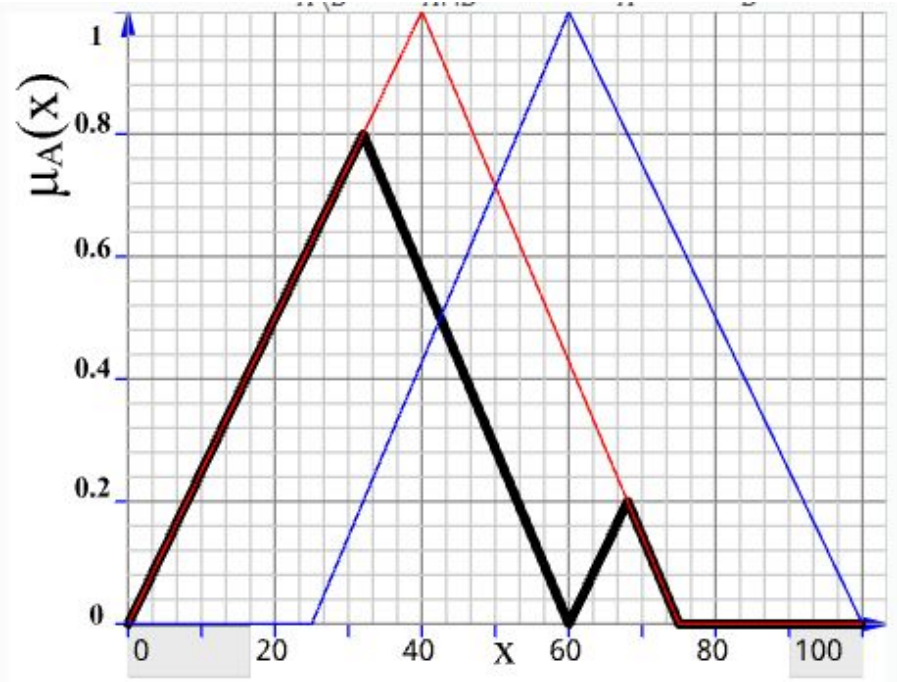
$$\forall x \in X \mu_A x = 1 - \mu_B x.$$

```
# дополнение
def complement(set_):
    complement_set = []
    for element in set_.get_set():
        complement_set.append([element[0], round(1 - element[1], 6)])
    return FuzzySet(complement_set)
```

Разность

Разность нечетких множеств A и B, заданных на универсальном множестве X, – это нечеткое множество A⧵B=A∩B с функцией принадлежности

$$\forall x \in X \mu_{A \setminus B} x = \mu_{A \cap \overline{B}} x = \min \mu_A x; 1 - \mu_B x.$$



Разность

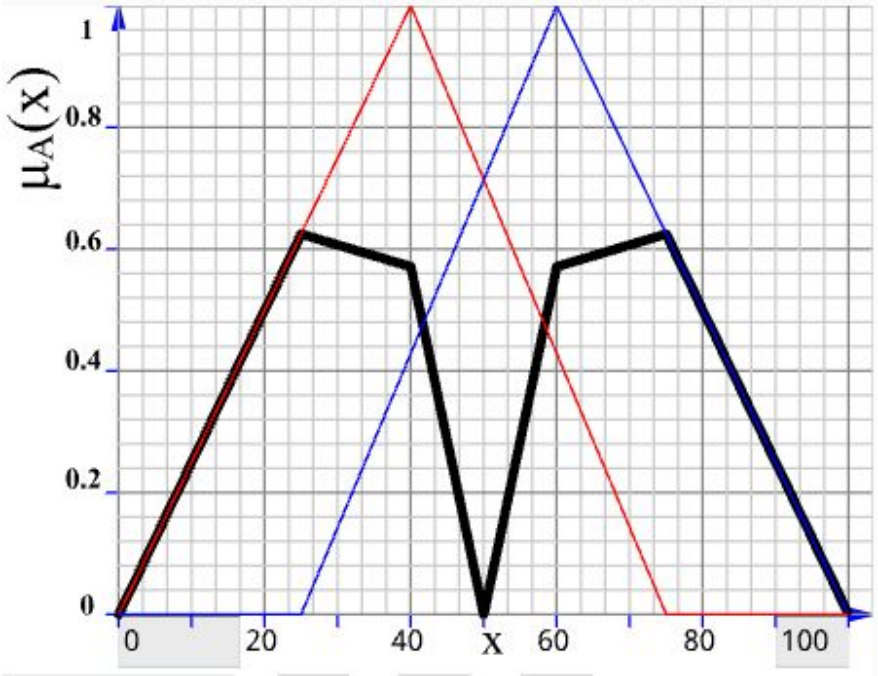
$$\forall x \in X \mu_{A \setminus B} x = \mu_{A \cap B} x = \min \mu_A x; 1 - \mu_B x.$$

```
# разность
def difference(set_a, set_b):
    difference_set = []
    comp_b = complement(set_b)
    for element in set_a.get_set():
        if element[0] in comp_b.get_elements():
            difference_set.append(
                [element[0], min(element[1], comp_b.get_set()[comp_b.get_elements().index(element[0])][1])])
        else:
            difference_set.append(element)
    for element in set_b.get_set():
        if element[0] not in set_a.get_elements():
            difference_set.append([element[0], 0])
    return FuzzySet(difference_set)
```

Симметрическая разность

Симметрическая разность нечетких множеств A и B, заданных на универсальном множестве X, – это нечеткое множество A−B с функцией принадлежности

$$\mu_{\tilde{A}\ominus\tilde{B}}(x) = |\mu_{\tilde{A}}(x) - \mu_{\tilde{B}}(x)|$$



Симметрическая разность

$$\mu_{\tilde{A} \ominus \tilde{B}}(x) = |\mu_{\tilde{A}}(x) - \mu_{\tilde{B}}(x)|$$

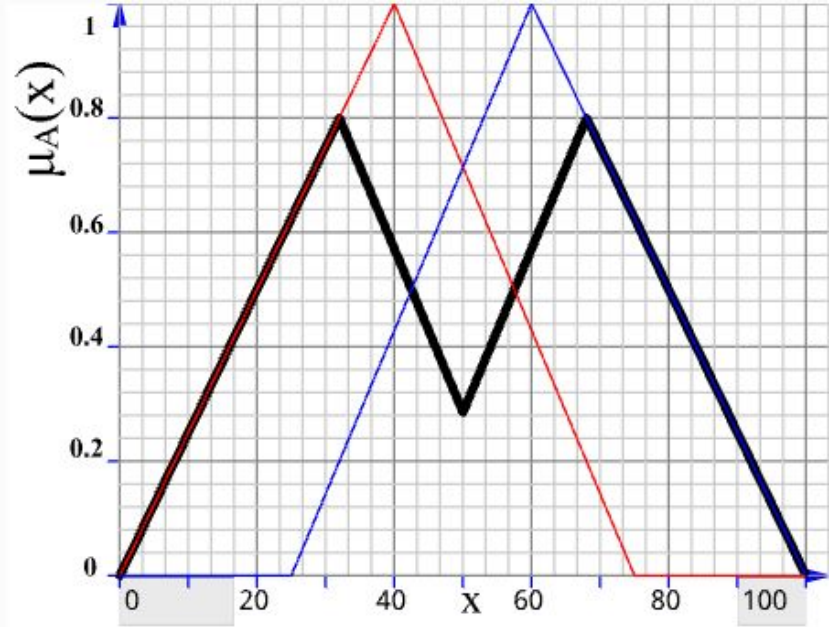
```
# симметрическая разность
def symmetrical_difference(set_a, set_b):
    s_difference_set = []
    for element in set_a.get_set():
        if element[0] in set_b.get_elements():
            s_difference_set.append([element[0], round(abs(element[1]
            - set_b.get_set()[set_b.get_elements().index(element[0])][1]), 6)])
        else:
            s_difference_set.append(element)
    for element in set_b.get_set():
        if element[0] not in set_a.get_elements():
            s_difference_set.append(element)
    return FuzzySet(s_difference_set)
```

Дизъюнктивная сумма

Дизъюнктивная сумма нечетких множеств A и B, заданных на универсальном множестве X, – это нечеткое множество $A \oplus B = A \setminus B \cup B \setminus A = A \cap \bar{B} \cup \bar{A} \cap B$

с функцией принадлежности

$$\forall x \in X \mu_{A \oplus B} x = \max\min \mu_A x; 1 - \mu_B x; \min 1 - \mu_A x; \mu_B x.$$



Дизъюнктивная сумма

$$A \oplus B = A \setminus B \cup B \setminus A = A \cap \bar{B} \cup \bar{A} \cap B$$

$$\forall x \in X \mu_{A \oplus B}(x) = \max\min(\mu_A(x); 1 - \mu_B(x); \min(1 - \mu_A(x); \mu_B(x)).$$

```
# дизъюнктивная сумма
def disjunctive_sum(set_a, set_b):
    return union(intersection(set_a, complement(set_b)), intersection(complement(set_a), set_b))
```

Тестирование операций

```
A:  [[30, 0.2]]
B:   [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Union:  [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Intersection:  [[30, 0.2]]
Complement A:  [[30, 0.8]]
Difference A - B:  [[30, 0.2], [10, 0], [9, 0], [8, 0], [12, 0]]
Difference B - A:  [[30, 0.8], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Symmetrical difference:  [[30, 0.6], [10, 1], [9, 1], [8, 0.88], [12, 0.5]]
Disjunctive sum:  [[30, 0.8]]
```