

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информационные системы в строительстве»

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ
ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

Методические указания к лабораторным работам

Ростов-на-Дону
ДГТУ
2022

УДК 004.8(075.8)

Составители: А.А. Ляпин

Интеллектуальные системы поддержки принятия решений:
метод. указания к лабораторным работам. – Ростов-на-Дону :
Донской гос. техн. ун-т, 2022. – 18 с.

Методические указания к лабораторным работам содержат краткую теорию по изучаемым вопросам, примеры разработки интеллектуальных систем поддержки принятия решений и индивидуальные задания.

Предназначены для обучающихся магистратуры по направлению подготовки 09.04.02 «Информационные системы и технологии» очного и заочного обучения.

Контрольная работа для студентов заочной формы обучения предполагает выполнение заданий 1, 2, 3 данных методических указаний.

УДК 004.8(075.8)

Печатается по решению редакционно-издательского совета
Донского государственного технического университета

Ответственный за выпуск зав. кафедрой «Информационные системы
в строительстве» д-р физ.-мат. наук, профессор А.А. Ляпин

В печать __.__.2022 г.
Формат 60×84/16. Объем 1,0 усл. п. л.
Тираж 60 экз. Заказ № ____

Издательский центр ДГТУ
Адрес университета и полиграфического предприятия:
344000, г. Ростов-на-Дону, пл. Гагарина, 1

© Донской государственный
технический университет, 2022

ОГЛАВЛЕНИЕ

1. НЕЧЕТКИЕ МНОЖЕСТВА И ОПЕРАЦИИ НАД НИМИ	4
1.1 Арифметические операции над нечеткими трапециевидными переменными	5
1.2 Операции нечеткой фильтрации и выбора.....	6
1.3 Задание 1.....	6
1.4 Задание 2.....	8
2. МЕХАНИЗМЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ SCIKIT-FUZZY	9
2.1 Задание нечетких множеств	9
2.2 Логические операции над нечеткими множествами	9
2.3 Операции фаззификации и дефаззификации.....	11
2.4 Задание 3.....	12
2.5 Пример дефаззификации нечетких значений	12
2.6 Пример нечеткого логического вывода.	13
ЛИТЕРАТУРА	18

1. НЕЧЕТКИЕ МНОЖЕСТВА И ОПЕРАЦИИ НАД НИМИ

Во многих областях знаний существуют ситуации, когда данные об объектах или явлениях являются слабо структурированными, не имеют количественного выражения, а законы, связывающие данные между собой, имеют описательный характер. Например, говоря о погоде, мы можем выразить ее состояние как «тепло», «холодно», «прохладно». А наше желание поехать за город, может быть определено как: «Если погода, будет теплой, а ветер несильным, то поездка обязательно состоится».

Для успешного принятия решения в таких областях необходимо оперировать с качественными данными и нечетко определенными понятиями. Впервые такие понятия были введены в статье «Fuzzy Sets» американского математика Лотфи Заде в 1965 году [1]. Им был создан специальный математический аппарат, называемый *теорией нечетких множеств* [2]. Основные идеи данного подхода состоят в построении зависимостей между нечетким понятием и набором значений числовой величины, определяющей его суть, в виде степени уверенности в том, что конкретное числовое значение принадлежит нечеткой величине. Отметим, что такое соответствие во многом является субъективным, как и сама теория нечетких множеств. Например, температуру воздуха 12 град С можно отнести к понятию «тепло», говоря о поездке за город, с уверенностью 0.6, где полная уверенность соответствует значению 1.

Нечетким множеством A на числовом множестве U называется совокупность пар $A = \langle \mu_A(u), u \rangle$,

где $\mu_A(u)$ - функция принадлежности нечеткого множества A ,

u - носитель нечеткого множества A , как подмножество U , для которого $\mu_A(u) > 0$.

$\mu_A(u)$ по смыслу соответствует степени уверенности в том, что некоторое значение u принадлежит нечеткому множеству.

Носители нечетких множеств могут быть *дискретными*, например $u = \{u_1, u_2, \dots, u_k\}$. Тогда A можно записать как

$$A = \left\{ \frac{u_1}{\mu_A(u_1)}, \frac{u_2}{\mu_A(u_2)}, \dots, \frac{u_k}{\mu_A(u_k)} \right\}$$

Или *непрерывными*. В этом случае u является числовым отрезком или объединением отрезков, а функция принадлежности задается аналитически или графически (рис. 1).

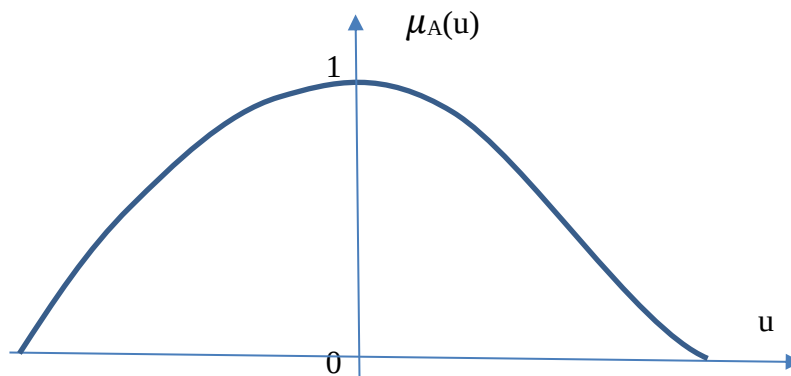


Рисунок 1. – Функция принадлежности

Следует отметить, что в большинстве случаев функции принадлежности строятся субъективно, например по результатам опроса экспертов или лиц, принимающих решение, поэтому они являются в некотором смысле «приближенными», т.е. не абсолютно адекватно отражающими явление или объект. Поэтому, очевидно, нужно выбирать такую функцию, с которой можно было бы как можно проще вести расчеты. Такими функциями являются *трапецевидные* (рис. 2) функции. Тогда $\mu_A(u)$ характеризуется четверкой $(\underline{m}, \overline{m}, \alpha, \beta)$. Как частный случай при $\underline{m} = \overline{m}$ имеем треугольную форму функции.

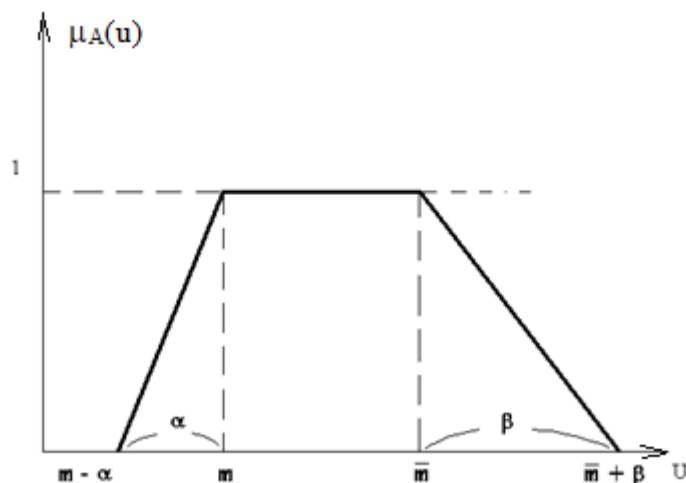


Рисунок 2. – Трапецевидная функция принадлежности

1.1 Арифметические операции над нечеткими трапецевидными переменными

Определение этих операции рассмотрим для случая двух нечетких переменных $\tilde{A}_1$ и $\tilde{A}_2$, которые заданы своими трапецевидными функциями принадлежности вида

$$\tilde{A}_1 = (\underline{m}_1, \overline{m}_1, \alpha_1, \beta_1),$$

$$\tilde{A}_2 = (\underline{m}_2, \overline{m}_2, \alpha_2, \beta_2).$$

Результатом операции будет также нечеткая переменная $A = (\underline{m}, \overline{m}, \alpha, \beta)$, которая также имеет трапецевидную функцию принадлежности, параметры которой определяются в зависимости от вида арифметической операции (табл.1)

Таблица 1

Тип операции	Зависимости параметров функций принадлежности
$A = \tilde{A}_1 (+) \tilde{A}_2$	$\underline{m} = \underline{m}_1 + \underline{m}_2, \quad \overline{m} = \overline{m}_1 + \overline{m}_2,$ $\alpha = \alpha_1 + \alpha_2, \quad \beta = \beta_1 + \beta_2$
$A = \tilde{A}_1 (-) \tilde{A}_2$	$\underline{m} = \underline{m}_1 - \overline{m}_2, \quad \overline{m} = \overline{m}_1 - \underline{m}_2,$ $\alpha = \alpha_1 + \beta_2, \quad \beta = \beta_1 + \alpha_2$
$A = \tilde{A}_1 (x) \tilde{A}_2$	$\underline{m} = \underline{m}_1 * \underline{m}_2, \quad \overline{m} = \overline{m}_1 * \overline{m}_2,$ $\alpha = \underline{m}_1 * \underline{m}_2 - (\underline{m}_1 - \alpha_1) * (\underline{m}_2 - \alpha_2),$ $\beta = (\overline{m}_1 + \beta_1) * (\overline{m}_2 + \beta_2) - \overline{m}_1 * \overline{m}_2$
$A = \tilde{A}_1 (/) \tilde{A}_2$	$\underline{m} = \underline{m}_1 / \overline{m}_2, \quad \overline{m} = \overline{m}_1 / \underline{m}_2,$ $\alpha = (\underline{m}_1 * \beta_2 + \overline{m}_2 * \alpha_1) / (\overline{m}_2^2 + \overline{m}_2 * \beta_2),$ $\beta = (\underline{m}_2 * \beta_1 + \overline{m}_1 * \alpha_2) / (\underline{m}_2^2 - \underline{m}_2 * \alpha_2)$

1.2 Операции нечеткой фильтрации и выбора

Для определения степени схожести двух нечетких переменных, или отнесения переменной Φ к одному из классов A_i , применяются операции нечеткой фильтрации или θ -отбора. Для решения данной задачи вводятся два показателя:

$P(A_i | \Phi) = \sup \min (\mu_\Phi(u), \mu_{A_i}(u))$ - это *возможность* того, что нечеткое множество Φ принадлежит значению A_i (мягкое решение).

$N(A_i | \Phi) = \inf \max (1 - \mu_\Phi(u), \mu_{A_i}(u))$ - это *необходимость* того, что нечеткое множество Φ принадлежит классу A_i (жесткое решение).

Для всех значений A_i мы находим пары (P_i, N_i) , а затем по максимальным значениям этих пар, находим принадлежность нечеткого множества к тому или иному значению атрибута A .

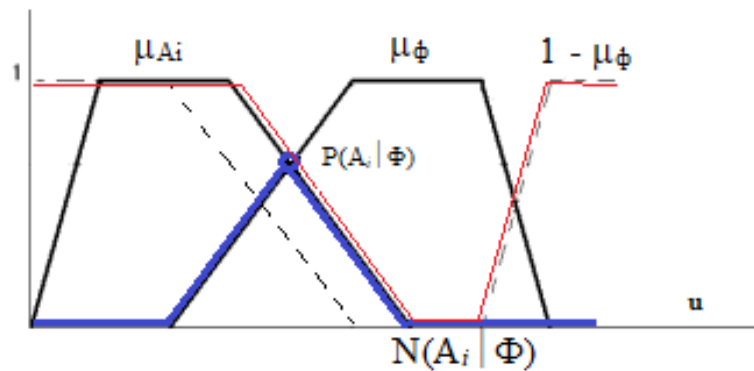


Рисунок 3.

Значения $P(A_i | \Phi)$, $N(A_i | \Phi)$ можно вычислить аналитически на основе формул принципа обобщения Заде. Пусть

$$\begin{aligned}\Phi &= (\underline{m}_1, \overline{m}_1, \alpha_1, \beta_1), \\ A_i &= (\underline{m}_2, \overline{m}_2, \alpha_2, \beta_2).\end{aligned}$$

Тогда

$$P(A_i | \Phi) = \min \left\{ \max \left(0, \min \left(1, 1 + \frac{\overline{m}_1 - \underline{m}_2}{\beta_1 + \alpha_2} \right) \right), \max \left(0, \min \left(1, 1 + \frac{\overline{m}_2 - \underline{m}_1}{\beta_2 + \alpha_1} \right) \right) \right\}; \quad (1)$$

$$N(A_i | \Phi) = \min \left\{ \max \left(0, \min \left(1, \frac{\underline{m}_1 - \underline{m}_2 + \alpha_2}{\alpha_1 + \alpha_2} \right) \right), \max \left(0, \min \left(1, \frac{\overline{m}_2 - \overline{m}_1 + \beta_2}{\beta_1 + \beta_2} \right) \right) \right\}. \quad (2)$$

1.3 Задание 1

Написать на языке Python [4] программу с определением класса нечетких трапециевидных чисел. Свойствами класса являются действительные числовые величины $\underline{m}$, $\overline{m}$, α , β . Методами – арифметические операции и процедуры вычисления возможности и необходимости по формулам (1), (2).

Проверить работоспособность программы на тестовых примерах.

Возможный фрагмент кода приведен ниже.

```
class ftrap(object):
    @property
    def m(self):
        return self.__m
```

```

@m.setter
def m(self,value):
    self.__m=value
@property
def M(self):
    return self.__M
@M.setter
def M(self,value):
    self.__M=value
@property
def alfa(self):
    return self.__alfa
@alfa.setter
def alfa(self,value):
    self.__alfa=value if(value>=0) else None
@property
def betta(self):
    return self.__betta
@betta.setter
def betta(self,value):
    self.__betta=value if(value>=0) else None

def __init__(self,m=0,M=0,alfa=0,betta=0):
    self.__m = m
    self.__M = M
    self.__alfa= alfa if(alfa>=0) else None
    self.__betta= betta if(betta >=0) else None

def fprint(self):
    print(f"({self.__m}, {self.__M}, {self.__alfa}, {self.__betta})")

def add(self,B):
    C.m = self.__m + B.m
    C.M = self.__M + B.M
    C.alfa = self.__alfa + B.alfa
    C.betta = self.__betta + B.betta
    return C

```

Здесь аналогично вышеприведенному методу сложения необходимо добавить методы `sub(self,B)`, `mul(self,B)`, `div(self,B)`, определяющие выполнение арифметических операций вычитания, умножения и деления с трапециевидными числами, а также методы вычисления возможности и необходимости.

```

if (__name__=="__main__"):
    A=ftrap(1,2,1,5)
    B=ftrap(3,5,1,2)
    C=ftrap()
    C=(A.add(B)).add(A)
    C.fprint()

```

1.4 Задание 2

Пусть в рамках составления проекта бюджета предприятия рассматриваются различные источники финансирования. Причем некоторые из них характеризуются неточностью оценки денежных сумм на день оценивания, а другие малой надежностью. Кроме того, из бюджета необходимо отдать долги, количество которых также неточно, так как зависит от того, потребует ли кредитор все или только часть в следующем финансовом периоде.

Источник А: финансирование обеспечивается, его сумма может изменяться от $40+10N$ до $100+10N$ млн. в зависимости от конъюнктуры, но с наибольшей вероятностью можно ожидать поступления в сумме от $50+10N$ до $90+10N$ млн.

Источник В: источник надежен и разумно полагать, что финансирование будет предоставлено и составит сумму от $100+10N$ до $110+10N$ млн.

Источник С: источник ненадежен, а если и даст, то не более $20+10N$ млн.

Долг D: плата за кредиты от $50+10N$ до $100+10N$ млн., но наиболее вероятна выплата $80+10N$ млн.

Здесь N – последняя цифра номера зачетки студента.

При этом любой из бюджетов может быть оценен как «малый» (A_1), «средний» (A_2) или «большой» (A_3). Т.е. мы имеем дело с лингвистической переменной «объем бюджета» ($\tilde{A}$), которая может принимать одно из трех нечетких значений (A_1, A_2, A_3).

Предположим, что функция принадлежности для каждого из значений была оценена следующим образом

$$A_1 = \text{«малые»} = (0, 50, 0, 50);$$

$$A_2 = \text{«средние»} = (80, 150, 20, 20);$$

$$A_3 = \text{«большие»} = (200, 250, 20, 20).$$

Задача - к какому из классов бюджетов можно отнести значение бюджета, полученного в примере. Т.е. необходимо определить является ли наш бюджет «малым», «средним» или «большим». Ответ сформулировать, опираясь на вычисление коэффициентов возможности и необходимости.

2. МЕХАНИЗМЫ НЕЧЕТКОГО ЛОГИЧЕСКОГО ВЫВОДА С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ SCIKIT-FUZZY

2.1 Задание нечетких множеств

Для задания нечеткого множества $A = \langle \mu_A(u), u \rangle$ в библиотеке Scikit-fuzzy необходимо определить отрезок U числовой оси, на котором задан носитель нечеткого множества u , а также функцию принадлежности $\mu_A(u)$.

U определяется в форме массива numpy с заданным шагом. Для использования функции принадлежности возможно ее задание как функции треугольного, трапециевидного вида, симметричной и несимметричной функции Гаусса, π - функции, ψ – функции и т.д. Информацию с описанием функций библиотеки Scikit-fuzzy можно найти на сайте: <https://pythonhosted.org/scikit-fuzzy/api/api.html> [5].

В качестве примера приведен фрагмент кода на Python с реализацией трапециевидной функции принадлежности и функции Гаусса. Результат представлен на рис. 4.

```
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
u = np.arange(0, 10.1, 0.1)
mfA = fuzz.trapmf(x, [2, 4, 6, 8])
mfB = fuzz.gaussmf(x, 7, 3)
plt.plot(u, mfA, u, mfB, "--")
```

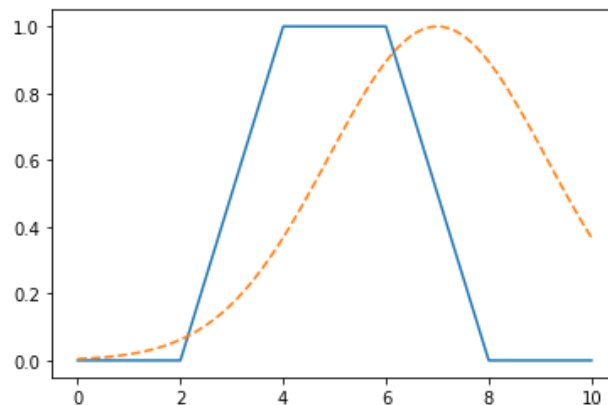


Рисунок 4. Графики функций принадлежности: трапециевидной и Гаусса

2.2 Логические операции над нечеткими множествами

Пусть A и B – два нечетких множества [3].

Равенство множеств A и B . Нечеткие множества A и B равны между собой, когда для всех элементов u_i обоих множеств выполняется условие $\mu_A(u_i) = \mu_B(u_i)$.

Логическая сумма. Логическая сумма (объединение) двух нечетких множеств A и B , $A \cup B$, определяется как наименьшее нечеткое множество, которое содержит как A , так и B :

$$\mu_{A \cup B}(u) = \max[\mu_A(u), \mu_B(u)].$$

Соответствующий оператор в библиотеке Scikit-fuzzy выглядит как:

```
u, mfC=fuzz.fuzzy_or(u, mfA, u, mfB)
```

Логическое произведение. Логическое произведение (пересечение) двух нечетких множеств A и B , обозначаемое $A \cap B$, определяется как наибольшее нечеткое множество, содержащееся одновременно в A и B .

$$\mu_{A \cap B}(u) = \min[\mu_A(u), \mu_B(u)].$$

В библиотеке Scikit-fuzzy используем команду (рис.5):

`u, mfD=fuzz.fuzzy_and(u, mfA, u, mfB)`

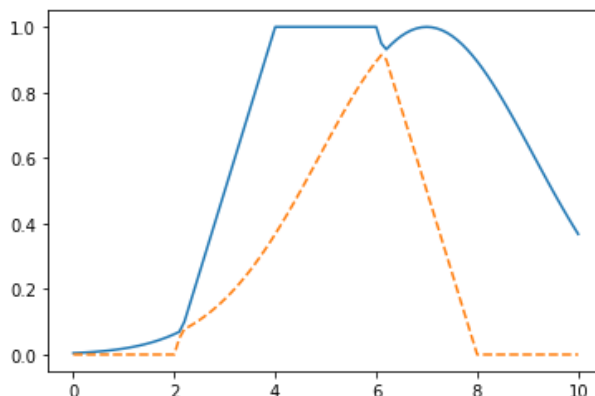


Рисунок 5. Графики функций принадлежности $\mu_{A \cup B}(u)$ (сплошная) и $\mu_{A \cap B}(u)$ (пунктирная линия)

Отрицание множества A. Соотношение для функций принадлежности элемента u к множеству A и его отрицанию $\bar{A}$ имеет вид:

$$\mu_{\bar{A}}(u) = 1 - \mu_A(u).$$

Нормализация множества. Операция нормализации множества $NORM(A)$ осуществляется по формуле:

$$\mu_{NORM(A)}(u) = \frac{\mu_A(u)}{\max\{\mu_A(u)\}}.$$

Концентрация. Операция концентрации $CON(A)$ записывается в виде:

$$\mu_{CON(A)}(u) = [\mu_A(u)]^2$$

Эта операция часто применяется при действиях с лингвистической переменной, в которых ее значение используется с термином “очень” (см. рис. 6)

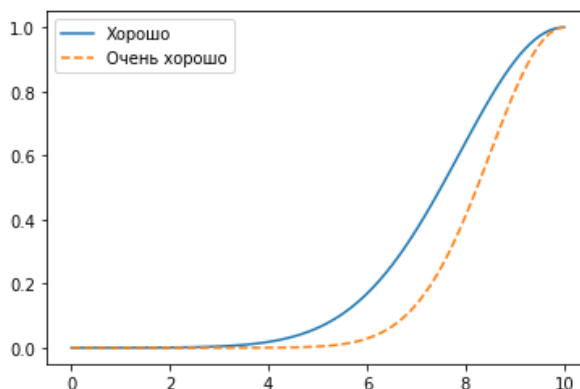


Рисунок 6.

Данная операция с массивами в Python может быть осуществлена обычным поэлементным умножением:

```
mfA = fuzz.gaussmf(x, 10, 3)
mfB = mfA*mfA
```

Для функции Гаусса это фактически означает уменьшение параметра σ , в нашем случае для нечеткого множества A равного 3.

Растяжение. Операция растяжения $DIL(A)$ записывается в виде:

$$\mu_{DIL(A)}(u) = [\mu_A(u)]^{1/2}$$

Лингвистическое значение этой операции формулируется как “примерно” или “почти”. Реализация в Python имеет вид (см. рис.7):

```
mfA = fuzz.gaussmf(x, 10, 3)
mfB = np.power(mfA,0.5)
plt.plot(u,mfA,u,mfB,"--")
plt.legend(['Хорошо', 'Почти хорошо'])
```

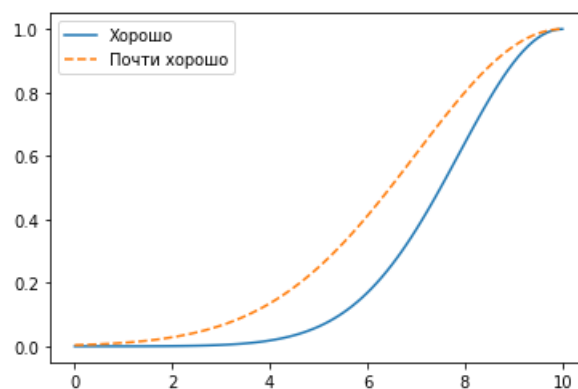


Рисунок 7.

2.3 Операции фаззификации и дефаззификации

Под операцией **фаззификации** будем понимать установление степени уверенности в принадлежности некоторого числового значения $u_i \in U$ нечеткому значению A лингвистической переменной X . Фактически необходимо определить ординату на графике функции принадлежности $\mu_A(u)$, соответствующую заданному числу u_i .

Например, определим степень уверенности в принадлежности числа 6 для значения A в приведенном выше примере с использованием метода `interp_membership` с точностью до 3 цифр после запятой.

```
u1=6
a1=fuzz.interp_membership(u,mfA,u1)
print(f"a1 = {a1:.3}")
```

Ответ: $a1 = 0.169$

Операция **дефаззификации** применяется в системах нечеткого вывода как процесс перехода от функции принадлежности для нечеткого значения лингвистической переменной к её четкому (числовому) значению.

2.4 Задание 3

1. С использованием библиотеки Scikit-fuzzy для универсального множества $u \in [-20, +40]$ определить функции принадлежности для значений лингвистической переменной, задающей температуру окружающего воздуха:

$A1 = \text{«холодно»}$, $A2 = \text{«прохладно»}$, $A3 = \text{«тепло»}$, $A4 = \text{«жарко»}$.

в виде интервальных чисел, треугольных чисел, трапециевидных чисел, гауссовых чисел, обобщенной функции колокола, π -функции, ψ -функции. Вывести графики соответствующих функций принадлежности.

2. Ввести произвольную температуру и определить степень принадлежности каждому значению лингвистической переменной с использованием функции `interp_membership`.

3. Произвести дефаззификацию каждого значения лингвистической переменной методами:

Centroid, bisector, mom, som, lom. Дать геометрическую интерпретацию каждого метода.

4. Провести арифметические операции со значениями лингвистической переменной: сложения, умножения, деления, вычитания, расширения, сужения.

5. Провести логические операции: конъюнкции, дизъюнкции, импликации, эквиваленции.

Для работы с библиотекой Scikit-fuzzy можно установить пакет Anaconda3, содержащий инструменты работы с Python. Воспользуемся ссылкой: <https://www.anaconda.com/products/distribution>

Загрузка библиотеки Scikit-fuzzy может быть осуществлена командой: `conda install -c conda-forge scikit-fuzzy` из командного окна пакета Anaconda3.

2.5 Пример дефаззификации нечетких значений

При вычислениях с нечеткой логикой нечеткий результат должен быть преобразован обратно в одно число. Существует несколько возможных методов дефаззификации, доступных через функцию `skfuzzy.defuzz`: Centroid, bisector, mom, som, lom.

Ниже приведен пример дефаззификации трапециевидного нечеткого значения различными методами:

```
import numpy as np
import matplotlib.pyplot as plt
import skfuzzy as fuzz
# Генерируем трапециевидную функцию принадлежности в диапазоне [0, 1]
x = np.arange(0, 5.05, 0.1)
mfx = fuzz.trapmf(x, [2, 2.5, 3, 4.5])

# Дефазифицируем эту функцию принадлежности пятью способами
defuzz_centroid = fuzz.defuzz(x, mfx, 'centroid') # Same as skfuzzy.centroid
defuzz_bisector = fuzz.defuzz(x, mfx, 'bisector')
defuzz_mom = fuzz.defuzz(x, mfx, 'mom')
defuzz_som = fuzz.defuzz(x, mfx, 'som')
defuzz_lom = fuzz.defuzz(x, mfx, 'lom')
```

```

labels = ['centroid', 'bisector', 'mean of maximum', 'min of maximum',
          'max of maximum']
xvals = [defuzz_centroid,
          defuzz_bisector,
          defuzz_mom,
          defuzz_som,
          defuzz_lom]
colors = ['r', 'b', 'g', 'c', 'm']
ymax = [fuzz.interp_membership(x, mfx, i) for i in xvals]

# Отображаем и сравниваем результаты
# дефаззификации с функцией членства

plt.figure(figsize=(8, 5))
plt.plot(x, mfx, 'k')
for xv, y, label, color in zip(xvals, ymax, labels, colors):
    plt.vlines(xv, 0, y, label=label, color=color)
plt.ylabel('Fuzzy membership')
plt.xlabel('Universe variable (arb)')
plt.ylim(-0.1, 1.1)
plt.legend(loc=2)

plt.show()

```

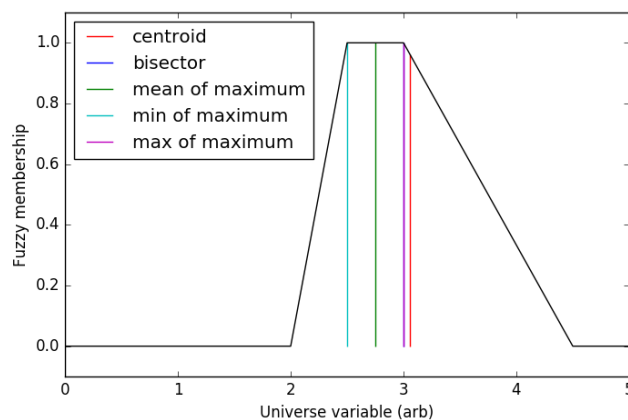


Рисунок 8.

2.6 Пример нечеткого логического вывода.

Данная реализация нечеткого логического вывода методом Мамдани проводит вычисления поэтапно простыми командами библиотеки Scikit-fuzzy шаг за шагом. Существует и интегрированный подход средствами библиотеки, связанный с использованием управляющей системы нечеткого логического вывода, здесь не рассматриваемый.

Рассмотрим задачу принятия решения для иллюстрации возможностей принципов нечеткой логики для генерации сложного поведения объектов на основе компактного, интуитивно понятного набора экспертных правил.

Входные переменные:

Ряд переменных влияет на принятие решения о том, сколько давать чаевых во время ужина. Рассмотрим два из них:

quality: Качество продуктов питания

service: Качество обслуживания

Выходная переменная - это сумма чаевых в процентах от счета: tip.

Ниже приведен код на Python решения данной задачи с использованием библиотеки Scikit-fuzzy.

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Генерируем значения нечетких лингвистических переменных
# * Качество продуктов и уровень обслуживания определим в диапазоне [0, 10]
# * Размер чаевых в диапазоне [0, 25] процентов от счета
x_qual = np.arange(0, 11, 1)
x_serv = np.arange(0, 11, 1)
x_tip = np.arange(0, 26, 1)

# Генерируем функции принадлежности нечетких значений
qual_lo = fuzz.trimf(x_qual, [0, 0, 5])
qual_md = fuzz.trimf(x_qual, [0, 5, 10])
qual_hi = fuzz.trimf(x_qual, [5, 10, 10])
serv_lo = fuzz.trimf(x_serv, [0, 0, 5])
serv_md = fuzz.trimf(x_serv, [0, 5, 10])
serv_hi = fuzz.trimf(x_serv, [5, 10, 10])
tip_lo = fuzz.trimf(x_tip, [0, 0, 13])
tip_md = fuzz.trimf(x_tip, [0, 13, 25])
tip_hi = fuzz.trimf(x_tip, [13, 25, 25])

# Визуализируем графики функций принадлежности на универсальных носителях
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))

ax0.plot(x_qual, qual_lo, 'b', linewidth=1.5, label='Bad')
ax0.plot(x_qual, qual_md, 'g', linewidth=1.5, label='Decent')
ax0.plot(x_qual, qual_hi, 'r', linewidth=1.5, label='Great')
ax0.set_title('Food quality')
ax0.legend()

ax1.plot(x_serv, serv_lo, 'b', linewidth=1.5, label='Poor')
ax1.plot(x_serv, serv_md, 'g', linewidth=1.5, label='Acceptable')
ax1.plot(x_serv, serv_hi, 'r', linewidth=1.5, label='Amazing')
ax1.set_title('Service quality')
ax1.legend()

ax2.plot(x_tip, tip_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_tip, tip_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_tip, tip_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Tip amount')
ax2.legend()
```

```

for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()

```

```
plt.tight_layout()
```

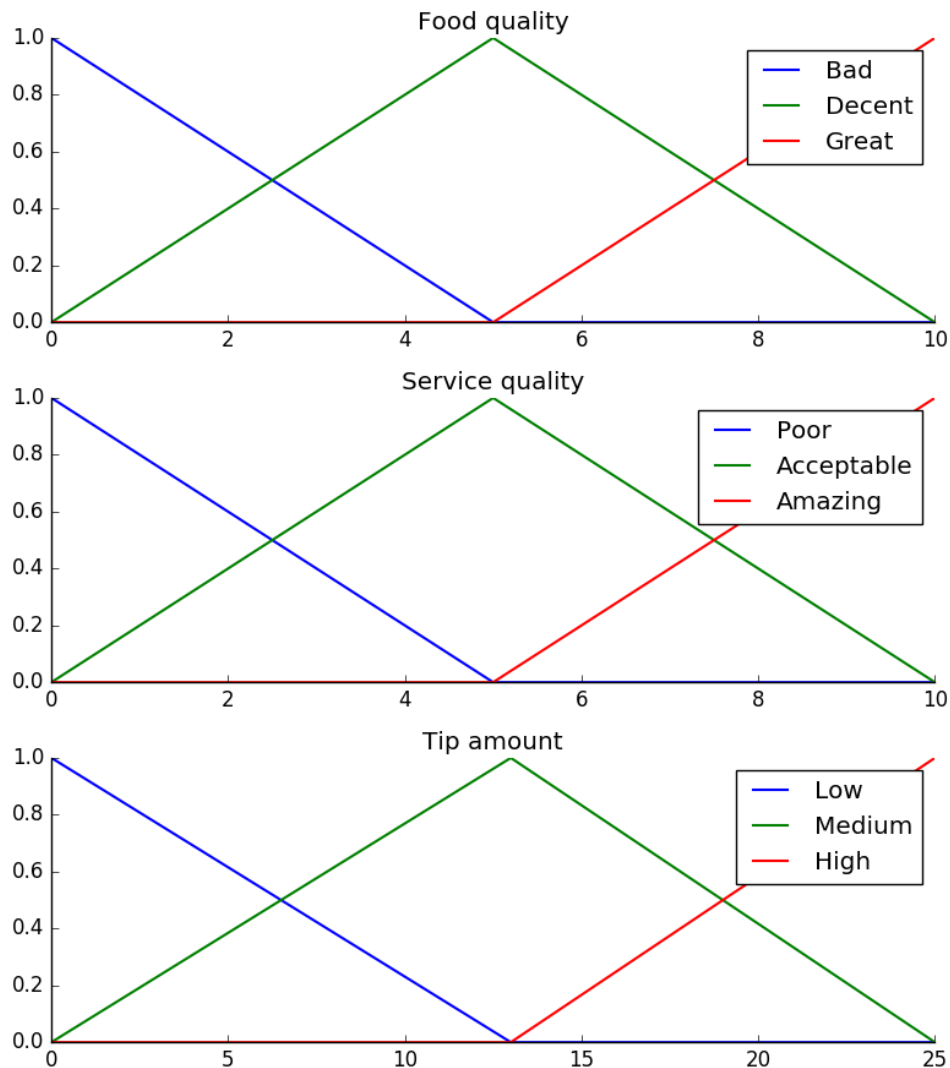


Рисунок 9.

Сформулируем **нечеткие правила**, определив нечеткую взаимосвязь между входными и выходными переменными. Для целей нашего примера рассмотрим три простых правила:

- Если еда плохая ИЛИ обслуживание плохое, то чаевые будут низкими
- Если услуга приемлема, то чаевые будут средними
- Если еда отличная ИЛИ обслуживание потрясающее, то чаевые будут высокими.

Большинство людей согласились бы с этими правилами, но правила нечеткие. Это та задача, в которой системы нечеткой логики имеют отличное применение.

С помощью нашей системы определим, каким был бы совет об уровне чаевых, если:

- Качество продуктов питания составило 6,5 баллов из 10.

- Обслуживание составило 9,8 баллов из 10.

Оценим степени уверенности исходных данных по их принадлежности различным значениям соответствующих лингвистических переменных:

```
qual_level_lo = fuzz.interp_membership(x_qual, qual_lo, 6.5)
qual_level_md = fuzz.interp_membership(x_qual, qual_md, 6.5)
qual_level_hi = fuzz.interp_membership(x_qual, qual_hi, 6.5)
```

```
serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, 9.8)
serv_level_md = fuzz.interp_membership(x_serv, serv_md, 9.8)
serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, 9.8)
```

Применим нечеткий вывод для каждого правила способом Мамдани.

```
# Правило 1
```

```
active_rule1 = np.fmax(qual_level_lo, serv_level_lo)
tip_activation_lo = np.fmin(active_rule1, tip_lo)
```

```
# Правило 2
```

```
tip_activation_md = np.fmin(serv_level_md, tip_md)
```

```
# Правило 3
```

```
active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
tip_activation_hi = np.fmin(active_rule3, tip_hi)
```

```
tip0 = np.zeros_like(x_tip)
```

```
# Визуализируем результат
```

```
tip0 = np.zeros_like(x_tip)
fig, ax0 = plt.subplots(figsize=(8, 3))
```

```
ax0.fill_between(x_tip, tip0, tip_activation_lo, facecolor='b', alpha=0.7)
ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.fill_between(x_tip, tip0, tip_activation_md, facecolor='g', alpha=0.7)
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, tip_activation_hi, facecolor='r', alpha=0.7)
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.set_title('Output membership activity')
```

```
for ax in (ax0,):
```

```
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
```

```
plt.tight_layout()
```

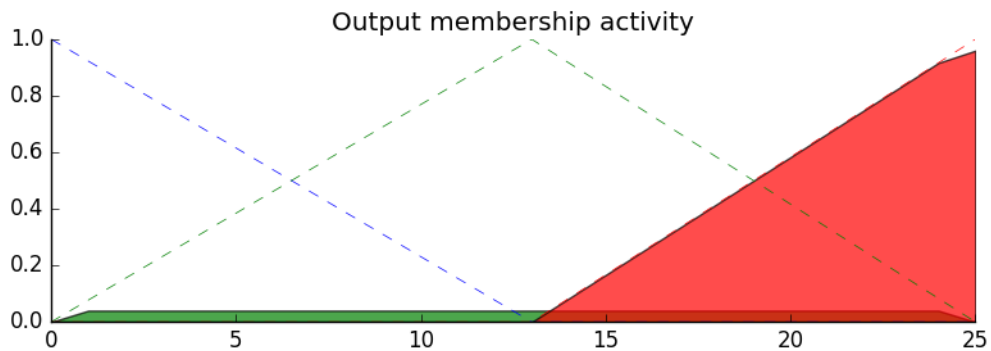



Рисунок 10.

В заключение проведем агрегирование правил и дефаззификацию полученного результата по уровню чаевых.

```
aggregated = np.fmax(tip_activation_lo,
                    np.fmax(tip_activation_md, tip_activation_hi))

# Вычисление четкого значения выходной переменной методом centroid.
tip = fuzz.defuzz(x_tip, aggregated, 'centroid')
tip_activation = fuzz.interp_membership(x_tip, aggregated, tip)
```

Визуализация

```
fig, ax0 = plt.subplots(figsize=(8, 3))

ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, aggregated, facecolor='Orange', alpha=0.7)
ax0.plot([tip, tip], [0, tip_activation], 'k', linewidth=1.5, alpha=0.9)
ax0.set_title('Aggregated membership and result (line)')

for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
```

plt.tight_layout()

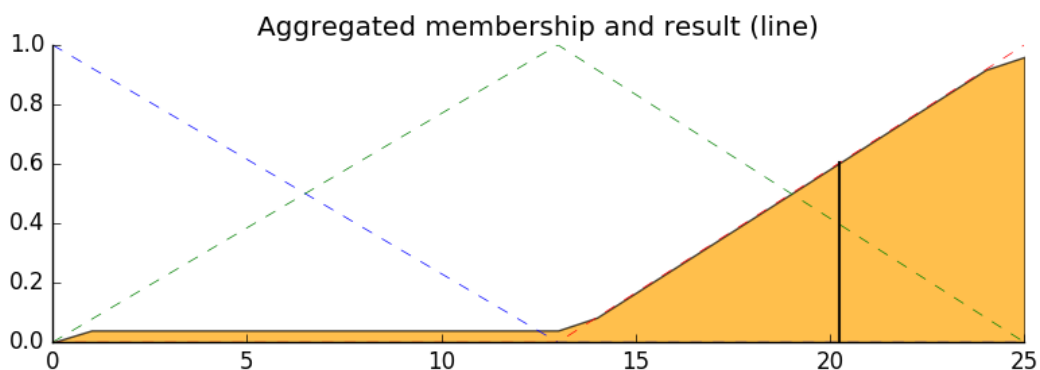


Рисунок 11.

В результате чаевые составляют 20,2%.

Заключительные замечания.

Отличительной особенностью нечетких систем является то, что они допускают сложное, интуитивно понятное поведение объектов исследования, основанное на конечном наборе правил, с минимальными вычислительными затратами. Обратите внимание, что наши функции принадлежности были грубыми, определяемыми только целыми числами, но функция `fuzz.interp_membership` позволяет увеличить точность вычислений, изменив шаг следования значений в массивах универсальных носителей для нечетких переменных, например:

```
x_qual = np.arange(0, 10.1, 0.1)
```

Литература

1. Zadeh L. A. Fuzzy sets // Information and Control. 1965. Т. 8, № 3. Р. 338-353.
2. Заде Л. Понятие лингвистической переменной и его применение к принятию приближенных решений. – М.: Мир, 1976. – 166 с.
3. Конышева Л.К., Назаров Д.М. Основы теории нечетких множеств: Учебное пособие. – СПб.: Питер, 2011. – 192 с.
4. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
5. Skfuzzy 0.2 docs. [Электронный ресурс] URL: <https://pythonhosted.org/scikit-fuzzy/api/api.html> (дата обращения: 24.01.2022)