Лабораторная работа

Тема: нечёткие системы

Цель работы: получение практических навыков программирования нечётких систем на языке Python с использованием библиотеки Skfuzzy .

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку Skfuzzy, NumPy и Matplotlib построить нечёткую базу знаний по варианту (совпадает с вариантом домашней работы, реализовать нечёткую базу знаний из домашней работы).

Работа заключается в построении:

- лингвистических переменных;
- нечётких продукций;
- поверхностей нечёткого вывода;
- использование нечёткой системы для получения конкретных результатов (не менее 3 прогонов с разными входными данными).

Общее количество лингвистических переменных должно быть не меньше 4, правил не менее 3, нечёткая база знаний должна быть полной.

Отчёт по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание, вариант
2. Описание предметной области и выбранных правил, в том числе каков результат работы системы, что является входными данными, в чем они измеряются и т.д..
3. Графики функций принадлежностей лингвистических переменных.
4. Поверхности нечёткого вывода.
5. Результаты нечёткого вывода (3 прогона).
6. Код.

Методические указания по использованию библиотеки Skfuzzy для построения нечётких систем

Схема построения нечёткого контроллера

Библиотека Skfuzzy содержит набор инструментов Fuzzy Logic для языка Python. Большая часть функциональности находится в подпакетах (см. Таблица 1)., но, как numpy, часть основных вынесено функций в базовое пространство имён (см. подробно https://pythonhosted.org/scikit-fuzzy/api/api.html).

Таблица 1 – Модули библиотеки Skfuzzy

Пакет (модуль)	**Описание**
control	Содержит инструменты для проектирования нечётких систем.
defuzzify	Содержит различные алгоритмы дефаззификации
cluster	Содержит нечёткий алгоритм кластеризации c-means
filters	Содержит инструменты для фильтрации данных
fuzzymath	Пакет нечёткой математики, содержащий основные математические операции для нечётких множеств и чётких переменных
image	Содержит основные операции для нечёткой логики на двумерных данных и изображениях.
intervals	Содержит операции для интервалов (сложение, вычитание, деление, умножение и масштабирование).
membership	Содержит генераторы нечёткой функции принадлежности

Для построения нечёткой системы (нечёткого контроллера) используют пакет Control и его классы (см. Рисунок 1).

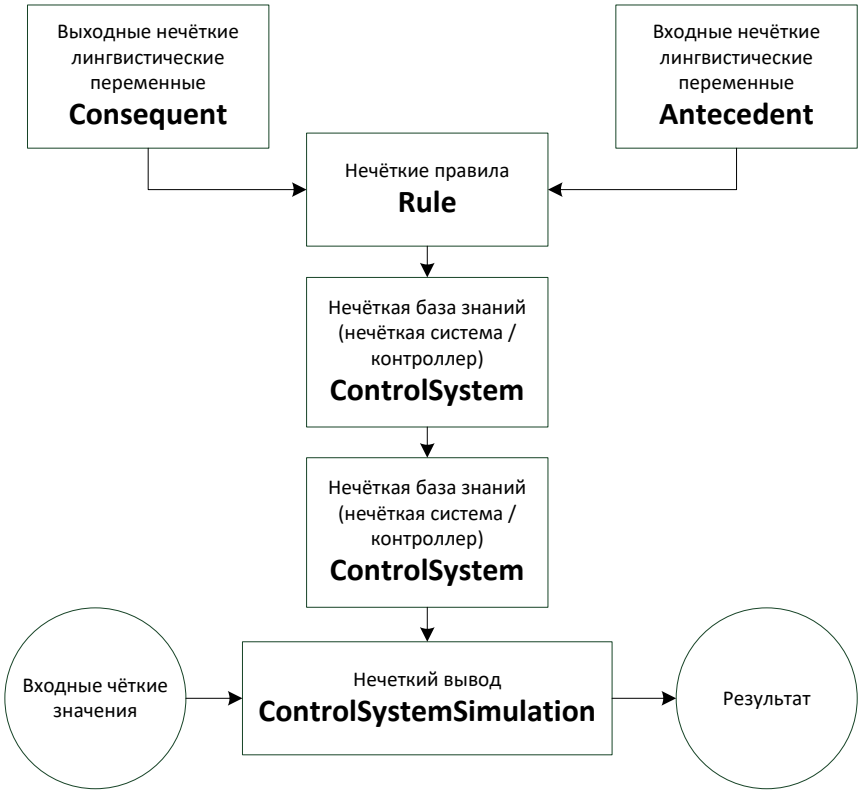


Рисунок 1 – Схема применения классов пакета Control

Определение лингвистических переменных

Определение лингвистической переменной

При описании лингвистической переменной необходимо определить входная она для задачи или выходная.

Входная лингвистическая переменная является антецедентом (предшествующим условием), а выходная – консеквентом (следствием).

Для входной лингвистической переменной используется метод `skfuzzy.control.Antecedent`(*universe*, *label*), для выходной - `skfuzzy.control.Consequent`(*universe*, *label*), где *label* – название переменной, *universe* – универсум (четкое множество, на котором задаётся нечёткая переменная), одномерный конвертируемый в NumPy массив.

Сам массив можно определит разными способами (см. например https://pyprog.pro/introduction.html).

Задание термов лингвистических переменные (нечётких переменных)

Использование пакета membership. Наиболее популярный способ определения функции нечёткой переменной - использование треугольной или трапециевидной функций, но существуют и другие варианты (Таблица 2).

Таблица 2 – Примеры методы генерации функций принадлежности (см. подробнее https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.membership.html)

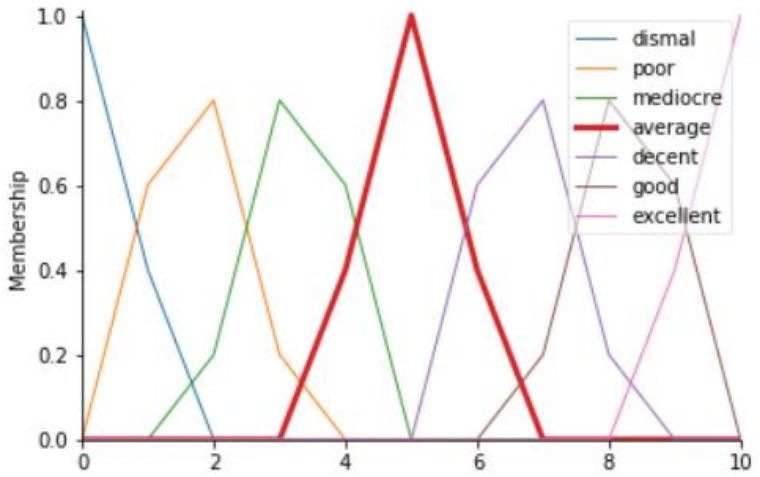
Метод	**Описание**	**Формула**
skfuzzy.membership.dsigmf(x, b1, c1, b2, c2)	Разница двух нечётких сигмоидальных функций принадлежности.	y = f1 - f2 f1(x) = 1 / (1. + exp[- c1 * (x - b1)]) f2(x) = 1 / (1. + exp[- c2 * (x - b2)])
skfuzzy.membership.gbellmf(x, a, b, c)	Генератор нечеткого членства обобщенной функции Белла.	y(x) = 1 / (1 + abs([x - c] / a) ** [2 * b])
skfuzzy.membership.piecemf(x, abc)	Кусочно-линейная функция принадлежности	y = 0, min(x) <= x <= a y = b(x - a)/c(b - a), a <= x <= b y = x/c, b <= x <= c
skfuzzy.membership.sigmf(x, b, c)	The basic sigmoid membership function generator.	y = 1 / (1. + exp[- c * (x - b)])
skfuzzy.membership.trapmf(x, abcd)	Трапециевидный генератор функций принадлежности.	См. Рисунок 2, б.
skfuzzy.membership.trimf(x, abc)	Треугольная функция принадлежности	См. Рисунок 2, с.

Используются методы генерации, например так:

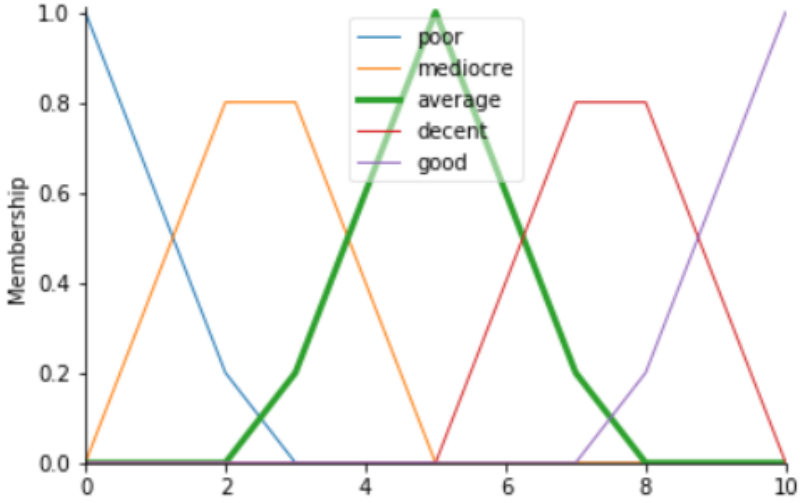
```
Имя_переменной['имя терма'] = skfuzzy.trimf(tip.universe, [0, 13, 25])
```

Автоматическое разбиение интервала. При построении функций принадлежности можно использовать автоматическую разбивку интервала на симметричные значения по 3, 5 или 7 термов (Рисунок 2). Имена при этом могут быть заданы автоматически: dismal, poor, mediocre, average, decent, good, excellent при разбиении на 7, - или пользователем. Для задания имён необходимо их определить в виде

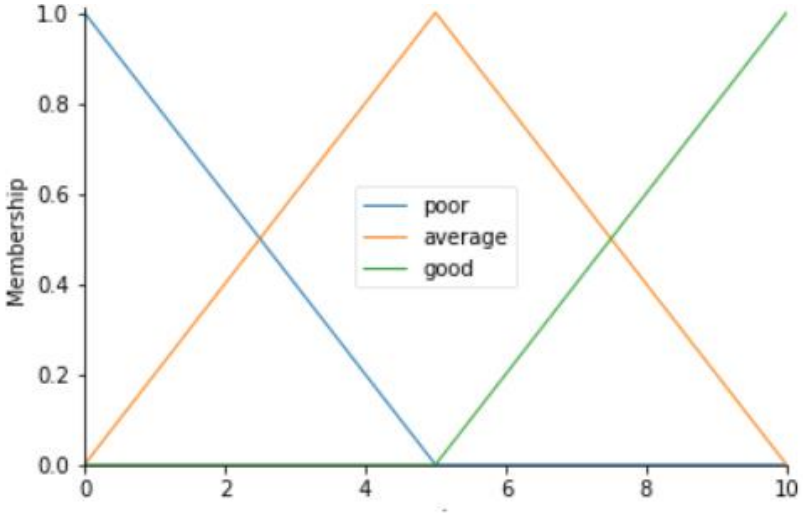
массива и подать как параметр automf, причем количество элементов массива должно быть также 3, 5, или 7 (например .automf(names= ['nb', 'ns', 'ze', 'ps', 'pb'])) .



а) 7 термов в лингвистической переменной



б) 5 термов в лингвистической переменной



с) 3 терма в лингвистической переменной

Рисунок 2 – Графики функций принадлежности при автоматической разбивке с помощью функции automf

Визуализация лингвистических переменных

Чтобы вывести график функций лингвистической переменной (Рисунок 2,) необходимо вызвать метод view(), для объектов классов `Antecedent` или `Consequent` (Имя_объекта.view()). Причём, если необходимо выделить конкретный терм, то необходимо указать имя терма (Имя_объекта['имя терма'].view()).

Определение нечёткой базы знаний (нечёткого контроллера)

Определение правила

Для задания правил используется Rule:

`skfuzzy.control.Rule`(*antecedent=None, consequent=None, label=None*)

Правило состоит из 2 частей: условия (*antecedent*) и следствия (*consequent*), - и имеет имя (*label*).

В условии и следствии могут использоваться логические операторы (см. Таблица 3)

Таблица 3 – Логические операторы в правилах

Логический оператор	**Обозначение**	**Пример**
Или	|	`.Rule(in_p1['poor']|in_p2['poor'], out_p['low'])`
И	&	`.Rule(antecedent=((in_p1['nb'] & in_p2['nb']) | (in_p1['ns'] & in_p2['nb']) | (in_p1['nb'] & in_p2['ns'])), consequent=out_p['nb'], label='rule nb')`
Не	~	`.Rule(~ in_p['poor'], out_p['high'])`

Правило можно визуализировать в виде графа с помощью метода .view().

Определение нечёткой базы / нечёткого контроллера

Нечёткая база состоит из нечётких правил. Для ее определения используют класс `skfuzzy.control.ControlSystem`(*rules=None*). Если все правила уже известны, то они передаются массивом, но правило в базы можно добавить и позже, используя метод `.addrule`(*rule*).

Использование нечёткой системы

Нечёткий вывод

После создания нечёткой базы (контроллера) его можно использовать для получения результата на конкретных значениях. Для этого необходимо подать на вход контроллера (модели) конкретные значения входных переменных (чёткие числа).

Для этого используется класс `skfuzzy.control.ControlSystemSimulation` и его метод `inputs(input_dict)`. Чтобы задать значение одной входной переменной требуется выполнить присвоение:

```
Имя_объекта.input['метка входной переменной'] = значение
```

Метод compute() класса skfuzzy.control.ControlSystemSimulation реализует нечеткий вывод.

Визуализация нечёткого вывода

Для визуализации результата можно использовать метод визуализации выходной нечеткой переменной с указанием в переменной sim объекта ControlSystemSimulation метод (пример view(sim=tipping)).

Для визуализации полной поверхности нечеткого вывда можно построить трехмерный график в случае зависимости одной выходной переменной от двух входных. Для этого можно использовать библиотеку Matplotlib.org (см. подробнее https://matplotlib.org). Ниже в примере приведён код, реализующий такую визуализацию.

Пример реализации нечёткой системы

Создаём нечёткую системы с 2 входными и одной выходной переменной, 3 правилам.

```
# pip install -U scikit-fuzzy
# подключаем библиотеку для работы с массивами
import numpy as np
# подключаем библиотеку для работы с нечёткими множествами
import skfuzzy as fuzz
from skfuzzy import control as ctrl

#библиотека для построения графика
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Создаем нечеткие переменные, которые будут частью условия (антецеденты)
# Antecedent (вход / датчик) переменная для нечёткой системы управления.
# skfuzzy.control.Antecedent(массив/список одномерный конвертируемый в NumPy,
метка / название)
# Для задания массива функцию arange(стартовое значение, конечное значение,
шаг),
# этот массив определяет универсум лингвистической переменной (массив четких
значений)

# Задаются 2 входные и 1 выходная лингвистическая переменная
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

# Разбиваем автоматически массив, для построения функции принадлежности,
можно выбрать вариант 3, 5 или 7 термов
quality.automf(3)
service.automf(3)

# Задаем выходную переменную через треугольную функцию
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])

# визуализируем переменные
quality['average'].view()
service.view()
tip.view()

# ЕСЛИ обслуживание было хорошим или качество еды было хорошим, ТОГДА чаевые
будут высокими.
# ЕСЛИ обслуживание было средним, ТО чаевые будут средними.
```

```
# ЕСЛИ обслуживание было плохим, а качество еды было плохим, ТОГДА чаевые
будут низкими.
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

rule1.view()
rule2.view()
rule3.view()
# Создаем базу из 3 правил
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
# Визуализируем
tipping_ctrl.view()

# Создаем модель расчёта
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
# Подаем на вход четкие числа
tipping.input['quality'] = 6.5
tipping.input['service'] = 9.8

# запускаем расчет
tipping.compute()

# Печатаем результат
print (tipping.output['tip'])
# выводим результат в виде графика
tip.view(sim=tipping)

# Строим трехмерную плоскость зависимости выходной переменной от 2 входных
# определяем значения по осям в виде массива
upsampled = np.arange(0, 26, 1)
# meshgrid создаем прямоугольную сетку из массива значений x и массив
значений y.
x, y = np.meshgrid(upsampled, upsampled)
# zeros_like() возвращает новый массив из нулей с формой и типом данных
указанного массива
z = np.zeros_like(x)

# вычисляем значения z в каждой точке
for i in range(26):
    for j in range(26):
        tipping.input['quality'] = x[i, j]
        tipping.input['service'] = y[i, j]
        tipping.compute()
        z[i, j] = tipping.output['tip']

# Строим по полученным значениям график
# определяем размер рисунка под график
fig = plt.figure(figsize=(25, 25))
# определяем трехмерность графика
ax = fig.add_subplot(111, projection='3d')

# создаем 3d поверхность
surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, cmap='viridis',
linewidth=0.4, antialiased=True)

# создаем контуры (проекции)
cset = ax.contourf(x, y, z, zdir='z', offset=-2.5, cmap='viridis', alpha=0.5)
cset = ax.contourf(x, y, z, zdir='x', offset=30, cmap='viridis', alpha=0.5)
cset = ax.contourf(x, y, z, zdir='y', offset=30, cmap='viridis', alpha=0.5)

# устанавливаем угол наклона графика и показываем
ax.view_init(50, 200)
```

Варианты заданий (дублируют домашнюю работу задача 2)

1. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи закупок (соотношения цены, качества, объема закупок и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
2. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузок спортсмена (соотношение нагрузок, физического состояния, потребляемых калорий и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
3. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (регулировка скорости с учетом передачи, погодных условий, интенсивности потока и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
4. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (управление рулем, газом, тормозом при въезде в гараж), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
5. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования теплоснабжения (соотношение среднесуточной температуры, ветра, размера здания и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
6. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования реверсного движения на волжском мосту (учитывать время, интенсивность потока, день недели и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
7. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора специй для блюда (соотношение количества и остроты специй, рецептуры, предпочтений едока, объема пищи и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
8. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора объема блюд (учитывать калорийность, вкусовые предпочтения, количество едоков и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
9. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подачи электроэнергии в условиях экономии (учет времени суток, типа помещений, количества людей, типа оборудования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

10. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора интенсивности занятий (учитывать начальный уровень подготовки, объем учебного материала, количество человек в группе, необходимый уровень усвоения и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
11. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи расчета потребления бензина (учитывать тип совершаемых маневров, уровень подготовки водителя, состояние автомобиля, тип автомобиля и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
12. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования системы орошения (учитывать время года, количество выпадающих озадков, вид орошаемой культуры и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
13. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи настройки аудиосистемы (мощность колонок, их количество, размер помещения, назначение установки и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
14. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора дозы снотворного (количество препарата, действие препарата, восприимчивость к выбранному препарату, цель и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
15. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи планирования объема производства продукции (с учетом возможной прибыли, необходимых ресурсов, платежеспособности населения, рынка сбыта и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
16. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования кондиционера (учитывать его мощность, объем помещения, температуру окружающей среды, необходимую температуру в помещении и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
17. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузки между компьютерами при использовании их в кластерах (учитывать характеристики компьютеров, их количество, количество параллельного кода, характеристики сети и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
18. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора складского помещения (учитывать площадь склада, количество и размеры продукции, удаленность от места производства и

точек реализации, свойства продукции и характеристики помещений и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

19. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора комплектующих для компьютера (учитывать цену, потребности пользователя, совместимость, сроки использования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
20. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи определения количества линий в службе поддержки (учитывать количество обслуживаемых клиентов, среднюю частоту обращения в службу одного клиента, среднее время обслуживания одной заявки, квалификацию персонала и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).