Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

Дисциплина «Системы искусственного интеллекта»

**Лабораторная работа 2**

**Нечеткая логика**

**Вариант №10**

Выполнил

Студент группы ИВТАСбд-41

Ведин В.А.

Проверил:

Старший преподаватель кафедры «ВТ»

Хайрулин И.Д.

Ульяновск

2025

# Задание

# На языке Python разработайте скрипт, позволяющий выполнить операцию импликации на заданных пользователем нечетких множеств с трапециевидными функциями принадлежности. Входными данными будут параметры функций принадлежности и четкие объекты для каждого из множеств. Выходными – результат импликации данных нечетких множеств. Причем, результат вывести через лингвистические переменные. Импликацию моделировать минимумом.

# Предметная область: Экология

# Уровень загрязнения воздуха: чисто, умеренное загрязнение, загрязнено, сильно загрязнено

# Уровень шума: тихо, средне, шумно, очень шумно

# Ход работы

# Задание параметров задачи

В данном пункте реализована функция задания параметров задачи — create\_mfs(), отвечающая за формирование функций принадлежности нечетких множеств. Её основная цель — предоставить пользователю возможность самостоятельно задать параметры функций, вместо использования заранее фиксированных значений. Это делает модель более гибкой и универсальной для разных сценариев.

Функция последовательно запрашивает у пользователя по четыре параметра для каждого уровня нечеткости — low, medium, high и very\_high. Введённые данные проходят несколько проверок: количество чисел должно быть ровно четыре; каждое из них обязано находиться в диапазоне от 0 до 100; кроме того, соблюдается логический порядок a ≤ b ≤ c ≤ d.

После успешной проверки параметры используются для построения трапециевидных функций с помощью метода fuzz.trapmf(). В случае некорректного ввода пользователю выводится сообщение об ошибке с пояснением, после чего предлагается повторить ввод.

Ниже, в листинге 1, представлен соответствующий фрагмент

Листинг 1. Задание параметров задачи

|  |
| --- |
| def create\_mfs(x: np.ndarray, name: str) -> dict[str, np.ndarray]:  mfs = {}  for level in ["low", "medium", "high", "very\_high"]:  while True:  try:  params = list(map(float, input(f"Введите 4 числа для {name} ({level}): ").split()))  if len(params) != 4:  raise ValueError("Должно быть ровно 4 числа.")  if not all(0 <= p <= 100 for p in params):  raise ValueError("Все числа должны быть от 0 до 100.")  if not (params[0] <= params[1] <= params[2] <= params[3]):  raise ValueError("Числа должны быть упорядочены: a <= b <= c <= d")  mfs[level] = fuzz.trapmf(x, params)  break  except ValueError as e:  print(f"Невверный ввод: {e}. попробуйте снова.")  return mfs |

# Функция импликации

# Функция implication() реализует ключевой этап нечеткого вывода — импликацию между входными и выходными нечеткими множествами. Она принимает на вход чёткие значения загрязнения и шума, а также наборы функций принадлежности для каждого из трёх рассматриваемых параметров: загрязнения воздуха, уровня шума и экологической опасности. Результатом работы функции является агрегированное нечеткое множество, отражающее итоговую степень экологического риска.

# На первом этапе вычисляются степени принадлежности введённых пользователем значений загрязнения и шума для всех уровней нечеткости (low, medium, high, very\_high). Это выполняется при помощи функции fuzz.interp\_membership(), которая определяет, насколько сильно конкретное значение принадлежит каждому из нечетких множеств.

# Далее задаётся база правил в виде словаря rules, связывающего комбинации уровней загрязнения и шума с соответствующим уровнем экологической опасности. Каждое правило обрабатывается по принципу минимизации (np.fmin): выбирается минимальная степень принадлежности между входными условиями, после чего результат ограничивается соответствующей функцией принадлежности выходного множества. Все полученные частные импликации объединяются в одно итоговое нечеткое множество с помощью операции максимизации (np.fmax.reduce).

# Функция импликации представлена ниже в листинге 2.

# Листинг 2. Функция имликации

|  |
| --- |
| def implication(pollution\_val: int, noise\_val: int,pollution\_mfs: dict, noise\_mfs: dict,eco\_mfs: dict) -> np.ndarray:mu\_pollution = {lvl: fuzz.interp\_membership(np.arange(0, 101, 1), mf, pollution\_val)for lvl, mf in pollution\_mfs.items()}mu\_noise = {lvl: fuzz.interp\_membership(np.arange(0, 101, 1), mf, noise\_val)for lvl, mf in noise\_mfs.items()}eco\_activations = []rules = {('low', 'low'): 'low',('low', 'medium'): 'medium',('low', 'high'): 'medium',('low', 'very\_high'): 'high',('medium', 'low'): 'medium',('medium', 'medium'): 'medium',('medium', 'high'): 'high',('medium', 'very\_high'): 'high',('high', 'low'): 'medium',('high', 'medium'): 'high',('high', 'high'): 'high',('high', 'very\_high'): 'very\_high',('very\_high', 'low'): 'high',('very\_high', 'medium'): 'high',('very\_high', 'high'): 'very\_high',('very\_high', 'very\_high'): 'very\_high',}x\_eco = np.arange(0, 101)for (p\_lvl, n\_lvl), eco\_lvl in rules.items():mu\_rule = np.fmin(mu\_pollution[p\_lvl], mu\_noise[n\_lvl])eco\_activations.append(np.fmin(mu\_rule, eco\_mfs[eco\_lvl]))aggregated = np.fmax.reduce(eco\_activations)return aggregated |

# Определение лингвистической метки

# Для удобства интерпретации результатов и вывода итогового уровня экологического риска был реализован механизм присвоения лингвистической метки конкретному числовому значению.

# Функция fuzzy\_label принимает три параметра: числовое значение, массив значений переменной и словарь с функциями принадлежности, соответствующими различным лингвистическим термам. Для каждого терма вычисляется степень принадлежности с помощью функции fuzz.interp\_membership. После этого выбирается та метка, у которой значение принадлежности максимально, и именно она возвращается в качестве итоговой лингвистической оценки.

# Код упомянутой функции представлен в листинге 3.

# Листинг 3. Определение лингвистической метки

|  |
| --- |
| def fuzzy\_label(value: float,x: np.ndarray,mfs: dict[str, np.ndarray]) -> str:memberships = {name: fuzz.interp\_membership(x, mf, value) for name, mf in mfs.items()}return max(memberships, key=memberships.get) |

# Интерпретация результатов и визуализация

# В данном пункте описывается процесс интерпретации результатов работы системы нечеткой логики и их визуализации. Основная функция main() отвечает за последовательный ввод параметров, обработку пользовательских значений и представление итогового результата в удобной форме.

# Сначала пользователю предлагается задать функции принадлежности для всех рассматриваемых множеств: загрязнения воздуха, уровня шума и экологической опасности. Для каждого множества используется функция create\_mfs(), обеспечивающая корректный ввод параметров и формирование трапециевидных функций принадлежности. После этого пользователь вводит конкретные значения загрязнения и шума, которые будут использоваться в процессе нечеткого вывода.

# Далее вызывается функция implication(), которая вычисляет агрегированное нечеткое множество экологической опасности на основе введённых данных и базы правил. Полученное нечеткое множество дефаззифицируется с помощью метода центроида, что позволяет получить одно численное значение степени экологического риска. Одновременно определяется словесная метка (low, medium, high, very\_high) для каждого параметра, обеспечивая удобное текстовое представление результата.

# Для наглядного отображения используется визуализация с помощью библиотеки matplotlib. Строятся графики всех функций принадлежности экологической опасности, а также отображается заполненная область, соответствующая результату импликации. График сохраняется в файл eco\_result.png

# Функция, упомянутая выше, представлена в листинге 4.

# Листинг 4. Интерпретация результатов и визуализация

|  |
| --- |
| def main() -> None:x = np.arange(0, 101, 1)print("=== Ввод функций принадлежности для загрязнения воздуха ===")polltion\_mfs = create\_mfs(x, "Загрязнение")print("\n=== Ввод функцй принадлежности для уровня шума ===")noise\_mfs = create\_mfs(x, "Шум")print("\n=== Ввод функций принадлежности для экологической опасности ===")eco\_mfs = create\_mfs(x, "Экологическая опасность")while True:try:pollution\_value = float(input("Введите уровень загрязнения воздуха (0-100): "))noise\_value = float(input("Введите уровень шума (0-100): "))if not (0 <= pollution\_value <= 100) or not (0 <= noise\_value <= 100):raise ValueErrorbreakexcept ValueError:print("Неверный ввод. Введите числа от 0 до 100.")result = implication(pollution\_value, noise\_value, polltion\_mfs, noise\_mfs, eco\_mfs)eco\_value = fuzz.defuzz(x, result, 'centroid')pollution\_label = fuzzy\_label(pollution\_value, x, polltion\_mfs)noise\_label = fuzzy\_label(noise\_value, x, noise\_mfs)eco\_label = fuzzy\_label(eco\_value, x, eco\_mfs)print(f"\nЗагрязнение ({pollution\_value:.1f}): {pollution\_label}")print(f"Шум ({noise\_value:.1f}): {noise\_label}")print(f"Экологическая опасность ({eco\_value:.1f}): {eco\_label}")plt.figure(figsize=(8, 4))for lvl, mf in eco\_mfs.items():plt.plot(x, mf, linestyle='--', label=lvl)plt.fill\_between(x, 0, result, color='orange', alpha=0.6, label="Импликация")plt.title(f'Результат импликации: экологическая опаность\n(Загрязнение {pollution\_value}, Шум: {noise\_value})')plt.xlabel('Экологическая опасность')plt.ylabel('Степень принадлежности')plt.legend()plt.savefig(f"eco\_result.png")if \_\_name\_\_ == '\_\_main\_\_':main() |

# Тестирование

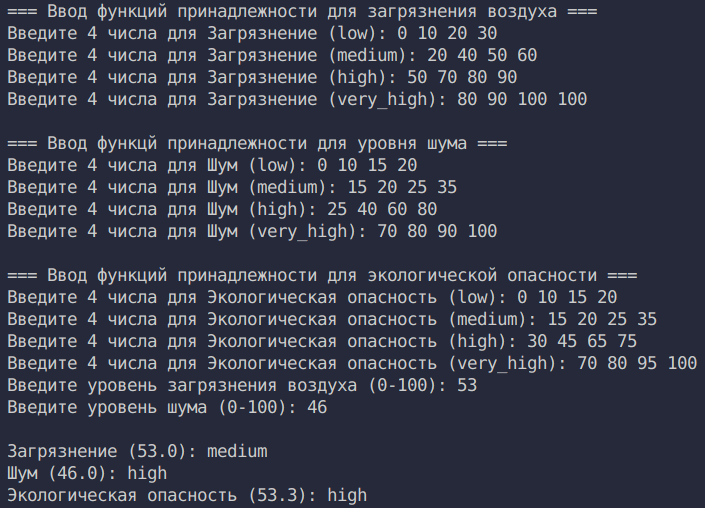


Рис. 1. Вывод результатов

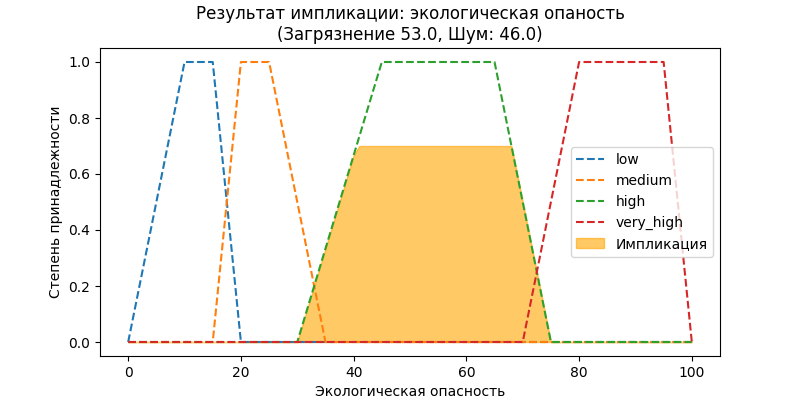


Рис. 2. График, отображающий результаты импликации

# Вывод

В лабораторной работе была создана модель нечёткого вывода для оценки экологического риска на основе уровней загрязнения воздуха и шума. Реализованы трапециевидные функции принадлежности, функция импликации с минимумом и присвоение лингвистических меток.

# Приложение.

# Ссылка на репозиторий с кодом лабораторной работы: <https://github.com/Tsaranchik/Artifical_Intelligence_Systems/tree/master/lab2>