

**Министерство науки и высшего образования РФ
ФГБОУ ВО «Ульяновский государственный университет»
Факультет математики, информационных и авиационных технологий**

Кафедра телекоммуникационных технологий и сетей

Липатова Светлана Валерьевна

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

для семинарских (практических) занятий, лабораторного практикума
и самостоятельной работы
по дисциплинам

«Интеллектуальные информационные системы» и «Системы искусственного интеллекта»

для студентов направлений

09.03.02 «Информационные системы и технологии»,

11.03.02 «Инфокоммуникационные технологии и системы»,

09.03.03 « Прикладная информатика»,

*02.03.03 «Математическое обеспечение и администрирование
информационных систем»*



УЛЬЯНОВСК
2019

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплинам «Интеллектуальные информационные системы» и «Системы искусственного интеллекта» / составитель: С.В. Липатова - Ульяновск: УлГУ, 2019 – 106 с.

Настоящие методические рекомендации предназначены для студентов направлений обучения 09.03.02 «Информационные системы и технологии», 11.03.02 «Инфокоммуникационные технологии и системы», 09.03.03 « Прикладная информатика», 02.03.03 «Математическое обеспечение и администрирование информационных систем». В работе приведены литература по дисциплине, темы дисциплины и вопросы в рамках каждой темы, рекомендации по изучению теоретического материала, контрольные вопросы для самоконтроля, задания для самостоятельной работы, задачи и упражнения для самостоятельной подготовки к семинарам или полностью самостоятельного освоения практических навыков, задания для лабораторного практикума и рекомендации по их выполнению.

Студентам всех форм обучения следует использовать данные методические рекомендации при подготовке к семинарам, самостоятельной подготовке, а также промежуточной аттестации по дисциплинам «Интеллектуальные информационные системы» и «Системы искусственного интеллекта».

Рекомендованы к введению в образовательный процесс

Учёным советом факультета математики, информационных и авиационных технологий
УлГУ

протокол № 2/19 от «19» марта 2019 г.

СОДЕРЖАНИЕ

ОБЩИЕ ВОПРОСЫ.....	5
РЕКОМЕНДАЦИИ ПО ОТДЕЛЬНЫМ ТЕМАМ ДИСЦИПЛИНЫ.....	6
<i>Тема 1. Философские вопросы искусственного интеллекта</i>	<i>6</i>
Основные вопросы темы.....	6
Рекомендации по изучению темы	6
Вопросы для самоподготовки	6
Контрольные тесты	6
<i>Тема 2. Подходы и направления исследований в области искусственного интеллекта</i>	<i>9</i>
Основные вопросы темы.....	9
Рекомендации по изучению темы	10
Вопросы для самоподготовки	10
Контрольные тесты	10
<i>Тема 3. Онтологии и Semantic Web</i>	<i>13</i>
Основные вопросы темы.....	13
Рекомендации по изучению темы	13
Вопросы для самоподготовки	13
Контрольные тесты	13
<i>Тема 4. Эволюционное моделирование.....</i>	<i>14</i>
Основные вопросы темы.....	14
Рекомендации по изучению темы	15
Вопросы для самоподготовки	15
Контрольные тесты:	16
Задания для практических занятий и самостоятельной работы	17
<i>Тема 5. Нечёткие вычисления.....</i>	<i>25</i>
Основные вопросы темы.....	25
Рекомендации по изучению темы	25
Вопросы для самоподготовки	25

Контрольные тесты	26
Задания для практических занятий и самостоятельной работы:	27
<i>Тема 6.Нейронные сети.</i>	33
Основные вопросы темы.....	33
Рекомендации по изучению темы	34
Вопросы для самоподготовки	34
Контрольные тесты	34
Задания для практических занятий и самостоятельной работы	37
ЛАБОРАТОРНЫЙ ПРАКТИКУМ	43
<i>Тема 3.Онтологии и Semantic Web</i>	43
Задание лабораторной работы.....	43
Методические указания по выполнению лабораторной работы	44
Варианты для выполнения лабораторной работы.....	50
<i>Тема 4.Эволюционное моделирование</i>	51
Задание лабораторной работы.....	51
Методические указания по выполнению лабораторной работы	51
Варианты для выполнения лабораторной работы.....	66
<i>Тема 5.Нечеткие вычисления.</i>	70
Задание лабораторной работы.....	70
Методические указания по выполнению лабораторной работы	70
Варианты для выполнения лабораторной работы.....	77
<i>Тема 6.Нейронные сети.</i>	80
Задание лабораторной работы.....	80
Методические указания по выполнению лабораторной работы	81
Варианты для выполнения лабораторной работы.....	103
РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ.....	105
Список рекомендуемой литературы	105
Программное обеспечение.....	106

ОБЩИЕ ВОПРОСЫ

В результате изучения дисциплин «Интеллектуальные информационные системы» и «Системы искусственного интеллекта» студенты должны:

- сформировать системное базовое представление, первичные знания, умения и навыки студентов по основам инженерии знаний и нейроинформатики,
- получить общее представление о прикладных системах искусственного интеллекта,
- получить представление о роли искусственного интеллекта и машинного обучения в развитии информатики в целом, а также, в научно-техническом прогрессе,
- подготовиться к применению концепций интеллектуальных систем при дальнейшем обучении,
- подготовиться к научной и практической деятельности в области разработки интеллектуальных систем в сферах прикладной деятельности.

Методические рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы по дисциплинам «Интеллектуальные информационные системы» и «Системы искусственного интеллекта» направлены на повышение эффективности освоения знаний, умений, навыков и компетенций, связанных с:

- онтологическим инжинирингом,
- эволюционным моделированием,
- искусственными нейронными сетями,
- нечёткими системами и др.

Методические рекомендации предлагают указания по всем темам дисциплин «Интеллектуальные информационные системы» и «Системы искусственного интеллекта». Методические рекомендации разбиты по темам и содержат набор вопросов для систематизации теоретического материала, полученного на лекционных занятиях, и самостоятельного изучения теории, вопросы (тесты) для текущего контроля на практических занятиях (семинарах), задачи для усвоения практических навыков. Для лабораторного практикума приведены задания, варианты и рекомендации по выполнению лабораторных работ.

Список литературы и информационного обеспечения, приведённый в конце методических указаний, может служить основой для изучения всех рассматриваемых тем. Дополнительная и учебно-методическая литература могут быть использованы обучающимися для закрепления изучаемого материала.

РЕКОМЕНДАЦИИ ПО ОТДЕЛЬНЫМ ТЕМАМ ДИСЦИПЛИНЫ

Тема 1. Философские вопросы искусственного интеллекта

Основные вопросы темы

1. Определения естественного и искусственного интеллекта.
2. Тест Тьюринга. Мысленный эксперимент «Китайская комната».
3. Теорема Геделя. Технологическая сингулярность.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 5.

Вопросы 2-3 изложены в учебнике [1] на с. 6-7.

Дополнительные материалы по теме представлены в [3] и [4] глава 1.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Что такое интеллект?
2. Что подразумевает термин «зима ИИ»? Сколько их было?
3. Что такое искусственный интеллект?
4. Какие виды ИИ выделяют?
5. Что собой представляет тест Тьюринга?
6. Что собой представляет мысленный эксперимент «Китайская комната»?
7. Зачем создавать ИИ?
8. Какие угрозы несёт ИИ?
9. Что подразумевает технологическая сингулярность?
10. Какие проблемы использования ИИ существуют?
11. Какая этика должна быть у ИИ?
12. Какой этике должен следовать исследователь ИИ?
13. Каковы общие тенденции развития науки?

Контрольные тесты

- 1) **Что из перечисленного ставило перед собой целью проверки интеллекта машины**

Выберите один ответ:

- а. Тест Тьюринга

- b. Премия Лёбнера
- c. Мысленный эксперимент «Китайская комната»
- d. Машина Тьюринга

2) Чем знаменателен 1964 год для искусственного интеллекта в России?

Выберите один ответ:

- a. Создана Ассоциация искусственного интеллекта
- b. Разработан метод обратный вывод Маслова
- c. Создан язык РЕФАЛ
- d. Нет правильного ответа

3) Кто считается «отцом» генетических алгоритмов?

Выберите один ответ:

- a. К. Де Йонг
- b. Д. Голдберг
- c. Нет правильного ответа
- d. Д. Холланд

4) В феврале 2011 года IBM решила протестировать свой IBM Watson в

Компьютер смог победить двух величайших чемпионов со значительным отрывом.

Выберите один ответ:

- a. викторина (аналог "Своя игра")
- b. шахматы
- c. покер
- d. го
- e. шашки

5) Гипотетический момент, по прошествии которого, по мнению сторонников данной концепции, технический прогресс станет настолько быстрым и сложным, что окажется недоступным пониманию- это технологическ....:

Выберите один ответ:

- a. сингулярность

- b. синергетика
- c. апокалипсис
- d. семантика

6) Что описано ниже?

Вопрос: Будь добр, напиши мне сонет на тему моста через Форт.

Ответ: Пожалуй, я пас. Я никогда не умел писать стихи.

В: Сложи 34957 и 70764.

О: (Компьютер выдерживает паузу в 30 секунд перед ответом) 105621.

В: Ты умеешь играть в шахматы?

О: Да.

В: [Описывает шахматную комбинацию]. Каков твой ход?

О: (Компьютер после паузы в 15 секунд) Ставлю тебе мат ладьей, приятель.

Выберите один ответ:

- a. Диалог с Алисой
- b. Протокол вопросов для прохождения теста Тьюринга
- c. Нет правильного ответа
- d. Описание возможного диалога для прохождения теста Тьюринга (статья «Вычислительная техника и интеллект», 1950)

7) Матч по игре Го между компьютерной программой AlphaGo, разработанной британской компанией Google DeepMind, и профессионалом 9 дана Ли Седолем был

Выберите один ответ:

- a. в 2016 и закончился победой AlphaGo
- b. в 2000 и закончился победой Ли
- c. в 2000 и закончился победой AlphaGo
- d. в 2016 и закончился победой Ли

8) Кто заложил основы теории нечетких множеств?

Выберите один ответ:

- a. И. Мамдани
- b. нет правильного ответа

- с. Б. Коско
- d. М. Блэк
- e. Л. Заде

9) Кто создал язык Lisp?

Выберите один ответ:

- a. Д. Маккарти
- b. В. Ф. Турчин
- c. Д. Робинсон
- d. Нет правильного ответа
- e. М. Минский

10) В чем суть закона Мура?

Выберите один ответ:

- a. Каждое следующее поколение компьютеров работает в 2,5 раза быстрее
- b. Не следует множить сущее без необходимости
- c. Если все слова языка или длинного текста упорядочить по убыванию частоты их использования, то частота n-го слова в таком списке окажется приблизительно обратно пропорциональной его порядковому номеру n
- d. 20% усилий дают 80% результата, а остальные 80% усилий — лишь 20% результата

Тема 2. Подходы и направления исследований в области искусственного интеллекта

Основные вопросы темы

1. Предыстория, история развития искусственного интеллекта как научного направления.
2. Нейрокибернетика и кибернетика «черного ящика».
3. История развития искусственного интеллекта в России.
4. Основные направления исследований в области искусственного интеллекта.
5. Свойства интеллектуальных информационных систем.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 6-9.

Вопрос 2 изложен в учебнике [1] на с. 13.

Вопрос 3 изложен в учебнике [1] на с. 6-9.

Вопрос 4 изложен в учебнике [1] на с. 13-15.

Вопрос 5 изложен в учебнике [1] на с. 22.

Дополнительный материал по вопросам представлен в [2] в разделах 2 и 3, [4] главе 1, 7.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Чем сильный ИИ отличается от слабого?
2. Какова основная идея нейрокибернетики?
3. Какова основная идея кибернетики чёрного ящика?
4. Какие задачи решает компьютерная лингвистика?
5. Какие виды анализа выполняют системы при машинном переводе?
6. Какие поколения роботов существуют?
7. Охарактеризуйте японский проект компьютер 5-го поколения?
8. На какие направления исследований делится искусственная жизнь?
9. Приведите примеры компьютерных игр с элементами ИИ и какие методы ИИ в них использовались?
10. Как реализуются творческие задачи в ИИ?

Контрольные тесты

1) Искусственная жизнь имеет следующие направления?

Выберите один или несколько ответов:

- а. влажная
- б. твердая
- в. сухая
- г. мягкая
- д. мокрая

2) Какой подход использует Булеву алгебру?

Выберите один ответ:

- a. логический
- b. структурный
- c. имитационный
- d. эволюционный

3) Экспертные знания активно используются в следующих направлениях?

Выберите один или несколько ответов:

- a. нет правильного ответа
- b. экспертные системы
- c. распознавание образов
- d. когнитивное моделирование

4) Сколько поколений роботов существует?

Выберите один ответ:

- a. 1
- b. 2
- c. 4
- d. 3
- e. 5

5) Какое из высказываний определяет основную идею нейрокибернетики?

Выберите один ответ:

- a. Не имеет значения, как устроено «мыслящее» устройство. Главное, чтобы на заданные входные воздействия оно реагировало так же, как человеческий мозг.
- b. Объект, способный мыслить, — это человеческий мозг. Поэтому любое «мыслящее» устройство должно каким-то образом воспроизводить его структуру.
- c. Нет ответа

6) Какие задачи решаются в рамках искусственного интеллекта?

Выберите один или несколько ответов:

- a. создание компьютерных игр
- b. принятие решений
- c. распознавание речи

- d. кодирование
- e. создание сред разработки информационных систем

7) К системам компьютерной лингвистики относятся:

Выберите один или несколько ответов:

- a. нет правильного ответа
- b. система распознавания речи
- c. машинный перевод
- d. система реферирования текстов
- e. система генерации музыки

8) Какое из направлений не придает значения тому, как именно моделируются функции мозга?

Выберите один ответ:

- a. нейрокибернетика
- b. нет правильного ответа
- c. кибернетика черного ящика

9) Принцип организации социальных систем используется в направлении?

Выберите один ответ:

- a. когнитивное моделирование
- b. многоагентные системы
- c. нейронные сети

10) Укажите направления искусственного интеллекта (выделить правильные ответы)

Выберите один или несколько ответов:

- a. телекоммуникационные технологии
- b. разработка естественно-языковых интерфейсов и машинный перевод
- c. машинное творчество
- d. технологии открытых систем
- e. обучение и самообучение
- f. распознавание образов

- g. представление знаний и разработка систем, основанных на знаниях
- h. геоинформационные технологии

Тема 3. Онтологии и Semantic Web

Основные вопросы темы

1. Определение онтологий.
2. Стек протоколов Semantic Web.
3. Основные элементы онтологий: классы, индивиды, свойства, аксиомы.
4. Использование правил в онтологиях.
5. Семантические машины вывода.

Рекомендации по изучению темы

Вопросы 1, 2 изложены в учебнике [5] в главе 1.

Вопрос 3 изложен в учебнике [5] в главе 2.

Вопрос 4 изложен в учебнике [5] в главе 3.

Вопрос 5 изложен в учебнике [5] в разделе 3.19.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Для чего предназначена онтология?
2. Как соотносятся протоколы Semantic Web с семиуровневой моделью OSI?
3. Из каких элементов строится онтология?
4. Какие запросы можно выполнить к онтологии?
5. Что из себя представляет триплет RDF?
6. Как объединяются онтология и правила?
7. Что такое слияние и выравнивание онтологий?
8. Какие существуют языки описания онтологий?
9. Что такое профиль OWL?
10. Что из себя представляет язык SWRL?

Контрольные тесты

1) Языком запросов к онтологии является:

Выберите один ответ:

- a. XML
- b. RDF
- c. OWL
- d. SPARQL

2) Тройка RDF это:

Выберите один ответ:

- a. Класс-связь-объект
- b. Класс-атрибут-метод
- c. Класс-объект-указатель
- d. Объект-отношение-субъект
- e. Объект-предикат-субъект
- f. Субъект-связь-объект

3) Онтологии связывают с концепцией

Выберите один ответ:

- a. Web 3.0
- b. Web 1.0
- c. Web 2.0
- d. Web 4.0

4) Профили OWL различаются

Выберите один ответ:

- a. по доступным функциям (базовый и расширенный вариант)
- b. по поддерживаемой модели представления знаний
- c. по поддерживаемой логике описания
- d. по времени создания (версия)

5) Ассоциирует объекты (классы) друг с другом

Выберите один ответ:

- a. индивиды (Individuals)
- b. свойства-связи (Object Property)
- c. свойства-характеристики (Datatype Property)

Тема 4. Эволюционное моделирование

Основные вопросы темы

1. Эволюционное моделирование.
2. Определение и основные понятия генетического алгоритма.
3. Операторы кроссовер, мутация, и инверсия. Фито-функция.

4. Методы отбора особей.
5. Виды генетического алгоритма.
6. Задачи, решаемые генетическим алгоритмом.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 62.

Вопрос 2 изложен в учебнике [1] на с. 65-66.

Вопросы 3, 4 изложены в учебнике [1] на с. 66-73.

Вопрос 5 изложен в учебнике [1] на с. 73-77.

Вопрос 6 изложен в учебнике [1] на с. 62.

Дополнительный материал по вопросам представлен в [4] главе 2.

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Какие задачи решаются в рамках эволюционных вычислений?
2. Какие принципы реализуются в эволюционных вычислениях?
3. Каковы достоинства эволюционных вычислений?
4. Каковы недостатки эволюционных вычислений?
5. Что такое генетическое программирование?
6. Что такое эволюционное программирование?
7. Что такое эволюционные стратегии?
8. Как выполняется представление программ в генетическом программировании?
9. Что такое Роевой интеллект?
10. Какие социальные / многоагентные алгоритмы существуют?
11. Какие основные шаги генетического алгоритма?
12. Какие типы существуют операторов ГА?
13. Какие стратегии формирования новой популяции существуют?
14. Чем рекомбинация отличается от кроссинговера?
15. Приведите примеры оператора кроссовера?
16. Каким может быть срок жизни особи?
17. Приведите примеры оператора мутации?
18. Что из себя представляет островная модель ГА?
19. Как выполняется турнирный отбор?
20. Что подразумевает принцип элитизма?

Контрольные тесты:

1) Какие понятия относятся к генетическим алгоритмам?

Выберите один или несколько ответов:

а. особь

б. функция активации

в. фенотип

г. ген

д. ДНК

е. нейрон

2) Какой оператор применен к особи (0001000 -> 0000000)?

Выберите один ответ:

а. скрещивания

б. нет правильного ответа

в. транслокация

г. инверсии

3) Из какого числа особей можно выбирать пару (второго родителя) для особи в островной модели (после случайного выбора)?

Выберите один ответ:

а. 8

б. t , выбирается случайным образом, чаще всего $t = 2$

в. m , где m – число особей в популяции

г. 4

д. $m-1$, где m – число особей в популяции

4) Какой оператор можно применить на этапе мутации?

Выберите один или несколько ответов:

а. нет правильного ответа

б. транслокации

в. кроссовер

г. инверсии

5) Какие виды генетического алгоритма подразумевают параллельную обработку?

Выберите один ответ:

а. гибридные алгоритмы

б. СНС

в. genitor

г. островная модель

е. нет правильного ответа

6) Какие виды отбора в генетических алгоритмах существуют?

Выберите один или несколько ответов:

- а. Дискретный отбор
- б. Поэтапный отбор
- в. Дуэльный отбор
- г. Рулетка
- д. Ранговый отбор
- е. Турнирный отбор

7) Какие методы относятся к направлению «Эволюционное моделирование»?

Выберите один или несколько ответов:

- а. Нейронные сети
- б. Генетические алгоритмы
- в. Эволюционное программирование
- г. Эвристическое программирование
- д. Метод группового учета аргументов

8) Какие бывают операторы генетического алгоритма?

Выберите один или несколько ответов:

- а. транслитерация
- б. конверсия
- в. мутация
- г. транслокация
- д. скрещивание
- е. кроссинговер

Задания для практических занятий и самостоятельной работы

Пример решения задачи

Задача. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, деленная на среднее значение суммы бит особей популяции; метод отбора – рулетка с принципом элитизма; оператор скрещивания – двухточечный кроссовер; оператор мутации – одиночная мутация.

Описание процесса решения. Для использования генетического алгоритма необходимо:

- 1) Определить набор признаков, характеризующие решения задачи оптимизации или моделирования. Определить фенотип, закодировать признаки (можно использовать код Грея).
- 2) Использовать последовательность шагов генетического алгоритма с соответствующими операторами.

Решение.

- 1) Фенотип (задаем десятичные значения случайным образом):

Признак	Двоичное значение признака	Десятичное значение признака	Код Грея
Признак 1	0011	3	0010
Признак 2	1100	12	1010
Признак 3	1110	14	1001
Признак 4	0111	7	0100
Признак 5	1001	9	1101

- 2) **1 шаг.** Формирование начальной популяции.

Определено пять признаков, пусть особь содержит любые 2 из них (два первых - значения первого критерия, три последних второго), случайным образом сгенерируем 10 особей, каждая особь длиной 8 бит:

Особь 1: 00111110

Особь 6: 00111110

Особь 2: 11001110

Особь 7: 11000111

Особь 3: 00111001

Особь 8: 00110111

Особь 4: 11001001

Особь 9: 10101010

Особь 5: 00110111

Особь 10: 01010101

- 2 шаг. Оценка особей популяции (используется фитнес-функция равная сумме бит в особи).

Особь	Сумма бит в особи	Приспособленность особи
1	5	1, 389
2	5	1, 389
3	4	1, 112
4	4	1, 112
5	5	1, 389

6	5	1, 389
7	5	1, 389
8	5	1, 389
9	4	1, 112
10	4	1, 112

Среднее значение суммы бит в популяции = 3,6.

3 шаг. Отбор (используется метод отбора – рулетка с принципом элитизма).
Строим рулетку (сектора пропорциональны приспособленности, рис. 7) и запускаем ее 8 раз (выбираем 4 пары, рис. 9):

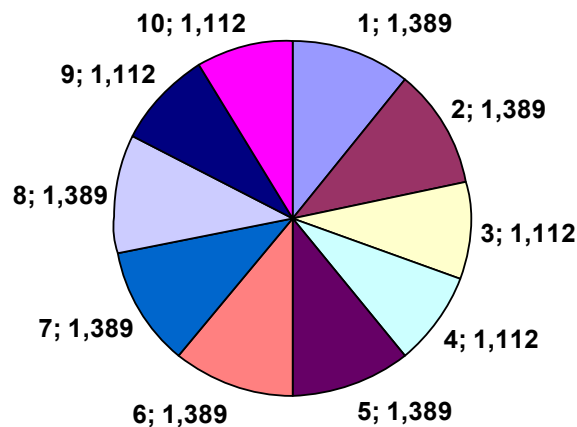


Рис. 7. Рулетка для задачи генетического алгоритма

Запуски рулетки (случайным образом):

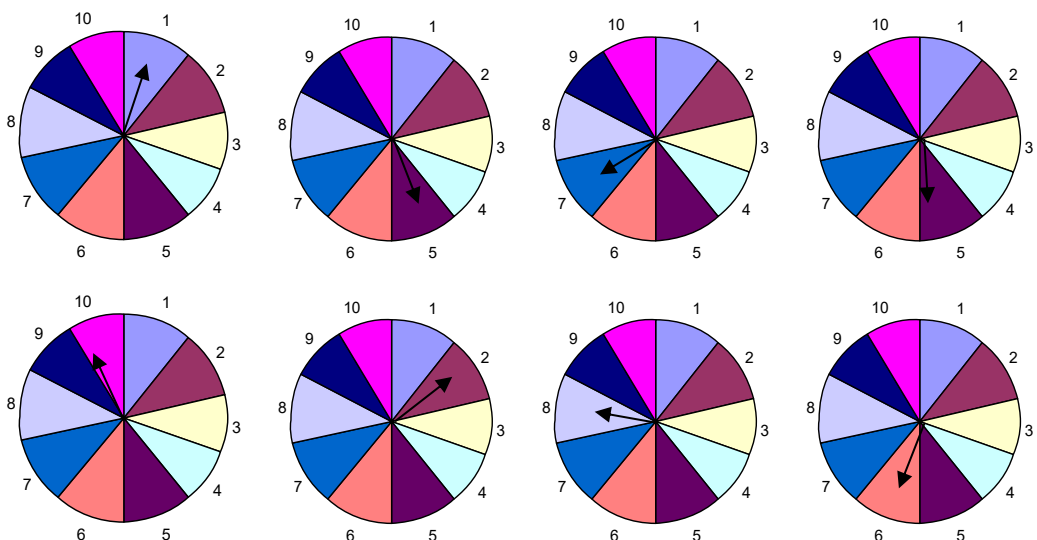


Рис. 8. Запуски рулетки для задачи генетического алгоритма

Таким образом, образовались следующие пары:

1 и 5, 7 и 5, 10 и 2, 8 и 6.

4 шаг. Скрещивание (используется оператор – двухточечный кроссовер).

Выбираем две точки разрыва (случайным образом, но числа должны различаться хотя бы на 2 и не быть равными 1 или длине особи): 2 и 5 и применяем оператор к выбранным парам особей:

Особь 1: 00|111|110 → *Особь 2.1:* 00|110|110
Особь 5: 00|110|111 *Особь 2.2:* 00|111|111

Особь 7: 11|000|111 → *Особь 2.3:* 11|110|111
Особь 5: 00|110|111 *Особь 2.4:* 00|000|111

Особь 10: 01|010|101 → *Особь 2.5:* 01|001|101
Особь 2: 11|001|110 *Особь 2.6:* 11|010|110

Особь 6: 00|111|110 → *Особь 2.7:* 00|110|110
Особь 8: 00|110|111 *Особь 2.8:* 00|111|111

5 шаг. Мутация (используется оператор – одноточечная мутация).

Определим вероятность мутации 30 % и бит – третий, подвергающийся мутации.

Особь	Случайное число	Особь	Мутированная особь
1	5	00111110	00011110
2	33	11001110	
3	67	00111001	
4	78	11001001	
5	90	00110111	
6	12	00111110	00011110
7	45	11000111	
8	53	00110111	
9	74	10101010	
10	29	01010101	01110101

6 шаг. Формирование новой популяции

Особь 2.1: 00110110 *Особь 2.7:* 00110110
Особь 2.2: 00111111 *Особь 2.8:* 00111111
Особь 2.3: 11110111 *Особь 2.9:* 00011110
Особь 2.4: 00000111 *Особь 2.10:* 00011110
Особь 2.5: 01001101 *Особь 2.11:* 01110101
Особь 2.6: 11010110 *Особь 2.12:* 00111110

1-8 – наследники, 9-11 – мутировавшие особи, 12 - сохраняем одну особь с максимальной приспособленностью – принцип элитизма.

7 шаг. Популяция достаточно разнообразна – нет признаков сходимости. Так как рассматривается лишь одна эпоха генетического алгоритма – выход из алгоритма.

Варианты задач

1. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, деленная на максимум суммы всех бит среди особей популяции; метод отбора – рулетка; оператор скрещивания – односточный кроссовер; оператор мутации – одиночная мутация.
2. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, деленная на минимум суммы всех бит среди особей популяции; метод отбора – турнирный отбор; оператор скрещивания – двухточечный кроссовер; оператор мутации – транслокация.
3. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – единица, деленная на минимум суммы всех бит среди особей популяции; метод отбора – ранговый отбор; оператор скрещивания – равномерный кроссовер; оператор мутации – инверсия.
4. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, умноженная на минимум суммы всех бит среди особей популяции; метод отбора – отбор усечением; оператор скрещивания – равномерный кроссовер; оператор мутации – односточная мутация.

5. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – единица, деленная на максимум суммы всех бит в особи в популяции; метод отбора – пропорциональный отбор; оператор скрещивания – одноточечный кроссовер; оператор мутации – инверсия.
6. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, деленная на количество бит в особи; метод отбора – рулетка с использованием принципа элитизма; оператор скрещивания – равномерный кроссовер; оператор мутации – инверсия.
7. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество бит в особи; метод отбора – пропорциональный с использованием принципа элитизма; оператор скрещивания – одноточечный кроссовер; оператор мутации – одноточечная мутация.
8. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – ранговый с использованием принципа элитизма; оператор скрещивания – одноточечный кроссовер; оператор мутации – одноточечная мутация.
9. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – турнирный с использованием принципа элитизма; оператор скрещивания – равномерный кроссовер; оператор мутации – транслокация.

10. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – отбор усечением с использованием принципа элитизма; оператор скрещивания – двухточечный кроссовер; оператор мутации – транслокация.
11. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, деленная на максимум суммы всех бит особи в популяции; метод отбора – рулетка; оператор скрещивания – двухточечный кроссовер; оператор мутации – одиночная мутация.
12. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на максимум суммы всех бит особи в популяции; метод отбора – турнирный отбор; оператор скрещивания – равномерный кроссовер; оператор мутации – инверсия.
13. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – единица, деленная на минимум суммы всех бит особи в популяции; метод отбора – ранговый отбор; оператор скрещивания – односточечный кроссовер; оператор мутации – инверсия.
14. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит, умноженная на минимум суммы всех бит особи в популяции; метод отбора – отбор усечением; оператор скрещивания – равномерный кроссовер; оператор мутации – транслокация.
15. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом,

начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – единица, деленная на максимум суммы всех бит среди особей популяции; метод отбора – пропорциональный отбор; оператор скрещивания – односточный кроссовер; оператор мутации – транслокация.

16. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество бит в особи; метод отбора – рулетка с использованием принципа элитизма; оператор скрещивания – односточный кроссовер; оператор мутации – транслокация.
17. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество бит в особи; метод отбора – пропорциональный с использованием принципа элитизма; оператор скрещивания – двухточечный кроссовер; оператор мутации – инверсия.
18. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – ранговый с использованием принципа элитизма; оператор скрещивания – равномерный кроссовер; оператор мутации – транслокация.
19. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом, начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – турнирный с использованием принципа элитизма; оператор скрещивания – равномерный кроссовер; оператор мутации – односточная мутация.
20. Описать функционирование одной эпохи генетического алгоритма на примере произвольной задачи (не менее пяти признаков закодировать случайным образом,

начальная популяция содержит не менее 10 особей). Использовать следующие параметры генетического алгоритма: фитнес-функция – сумма всех бит особи, деленная на количество особей в популяции; метод отбора – отбор усечением с использованием принципа элитизма; оператор скрещивания – двухточечный кроссовер; оператор мутации – одноточечная мутация.

Тема 5. Нечёткие вычисления.

Основные вопросы темы

1. Теория нечётких множеств.
2. Понятие нечеткого множества.
3. Функция принадлежности.
4. Операции с нечеткими множествами.
5. Нечеткие отношения.
6. Лингвистическая переменная.
7. Нечеткие высказывания.
8. Нечеткая импликация.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 79-81.

Вопрос 2-4 изложен в учебнике [1] на с. 81-84.

Вопросы 6-8 изложены в учебнике [1] на с. 84-91.

Дополнительные материалы по теме представлены в [6].

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Как задаётся нечёткое множество?
2. Какие значения может принимать функция принадлежности?
3. Как определяется нечеткая переменная?
4. Как определяется лингвистическая переменная?
5. Какие виды нечётких высказываний существуют?
6. Что такое фаззификация и дефаззификация?
7. Какие методы применяются на этапе аккумуляции?
8. Какие схемы нечёткого вывода существуют?
9. Каковы достоинства и недостатки нечётких систем?

10. Какие методы определения функций принадлежности существуют?

Контрольные тесты

1) **Множество точек, для которых функция принадлежности равна 1, называется?**

Выберите один ответ:

- a. Нет правильного ответа
- b. альфа -срезом
- c. носителем
- d. ядром

2) **Объединение нечетких множеств А и В определяет какая из формул?**

1) $\max\{m_B(u), m_A(u)\}$

2) $\min\{m_B(u), m_A(u)\}$

3) $1-m_A(u)$

4) $1-\min\{m_B(u), m_A(u)\}$

Выберите один или несколько ответов:

- a. 1
- b. нет правильного ответа
- c. 3
- d. 2
- e. 4

3) **Пусть $\mu_A(u)$, $\mu_B(u)$ – функции принадлежности нечетких множества А и В на универсальном множестве U. Пусть также С – нечеткое множество с функцией принадлежности $\mu_C(u)$, которое является объединением А и В. Определить значение принадлежности $u \in U$ нечеткому множеству С, если $\mu_A(u)=0,5$ и $\mu_B(u) = 0$?**

1) $m_C(u) = \max\{m_B(u), m_A(u)\} = 0,5$

2) $m_C(u) = \min\{m_B(u), m_A(u)\} = 0$

3) $m_C(u) = 1-\min\{m_B(u), m_A(u)\} = 1$

4) Нет правильного ответа

Выберите один ответ:

- a. 4
- b. 2
- c. 3
- d. 1

4) Функция принадлежности может принимать значения..?

Выберите один ответ:

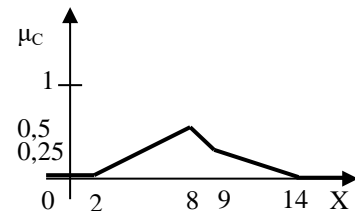
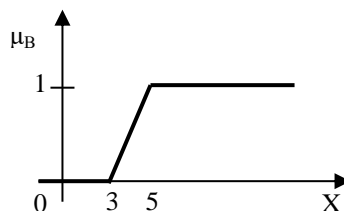
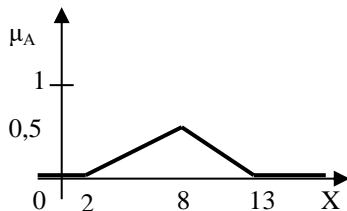
- a. $[0, 1]$
- b. $[0, \infty]$
- c. Нет правильного ответа
- d. $[-\infty, +\infty]$

Задания для практических занятий и самостоятельной работы:

Пример решения задачи

Задача. Дано 3 нечетких множества А, В, С (заданы их функции принадлежности).

Построить функцию принадлежности нечеткого множества $D = \bar{A} \cap (A \cup C \cup B)$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

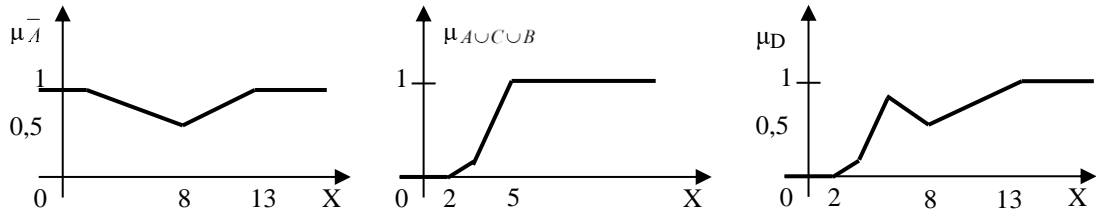


Описание процесса решения. Для построения функции принадлежности нового множества необходимо:

- 1) Определить последовательность выполнения операций в формуле.
- 2) Построить на отдельных графиках промежуточные множества, согласно определенной последовательности действий. Свести промежуточные множества на одном графике и определить итоговую функцию принадлежности.
- 3) Используя определенный в задаче метод, определить аналитически степень принадлежности элемента, входящего в ядро итогового множества.
- 4) Проверить аналитические вычисления по построенному графику функции принадлежности.

Решение.

- 1) Множество $D = \bar{A} \cap (A \cup C \cup B)$, значит, последовательность операций будет следующей: \bar{A} , $A \cup C \cup B$, $\bar{A} \cap (A \cup C \cup B)$.
- 2) Построим согласно этой последовательности операций графики функций принадлежности:



- 3) Ядро множества D состоит из элементов из интервала $(2,13)$. Выберем элемент 8.

$$\mu_A(8) = 0.5;$$

$$\mu_B(8) = 1;$$

$$\mu_C(8) = 0.5;$$

$$\mu_{\bar{A}}(8) = 1 - \mu_A(8) = 1 - 0.5 = 0.5;$$

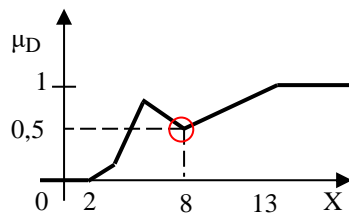
$$\mu_{\bar{C}}(8) = 1 - \mu_C(8) = 1 - 0.5 = 0.5;$$

$$\mu_{A \cup C}(8) = \min\{1, \mu_C(8) + \mu_A(8)\} = \min\{1, 0.5 + 0.5\} = 1;$$

$$\mu_{A \cup C \cup B}(8) = \min\{1, \mu_{A \cup C}(8) + \mu_B(8)\} = \min\{1, 1 + 1\} = 1;$$

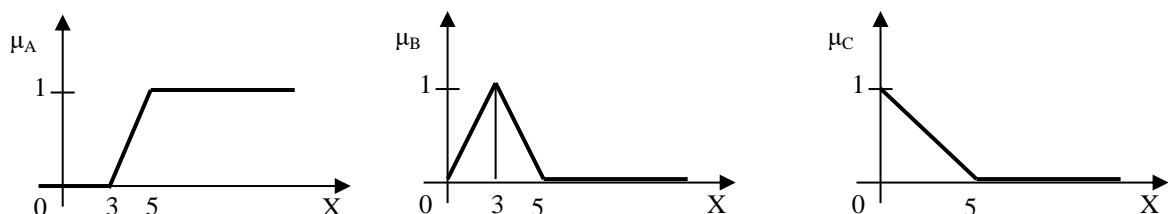
$$\mu_{\bar{A} \cap (A \cup C \cup B)}(8) = \max\{0, \mu_{\bar{A}}(8) + \mu_{A \cup C \cup B}(8) - 1\} = \max\{0, 0.5 + 1 - 1\} = 0.5.$$

- 4) $\mu_D(8) = 0.5;$

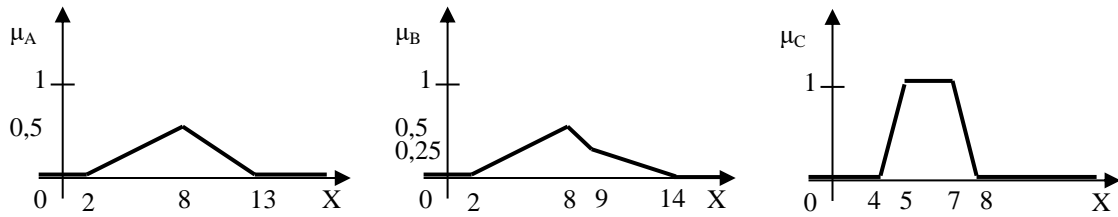


Варианты задач

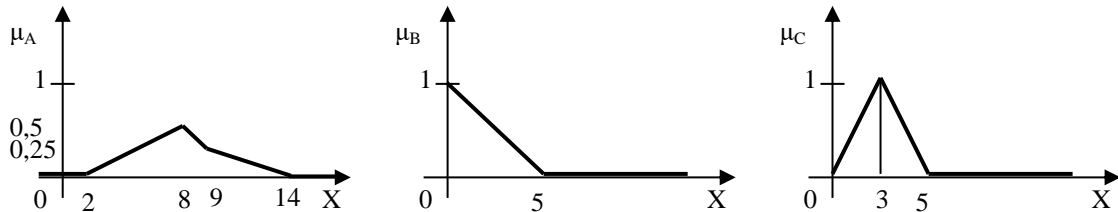
- 1) Дано 3 нечетких множества A , B , C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D , используя максиминный способ.



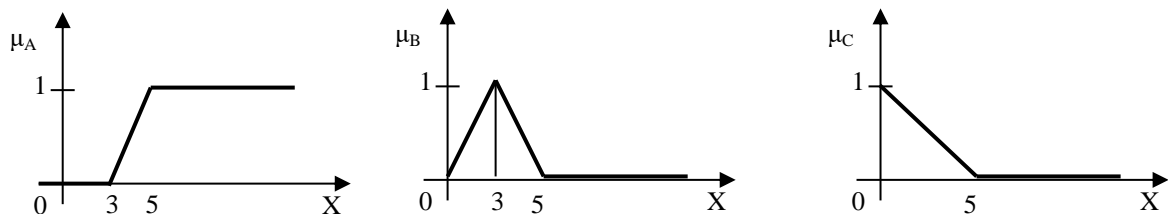
- 2) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



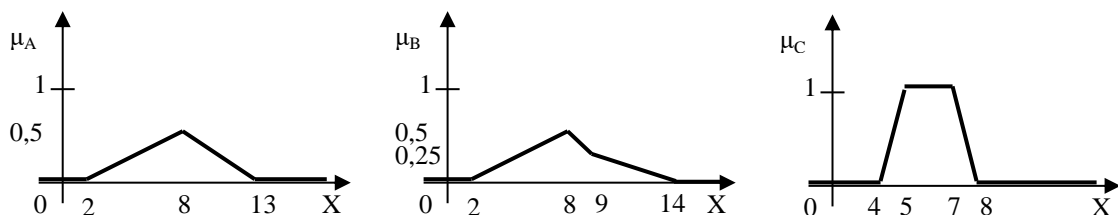
- 3) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



- 4) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \bar{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

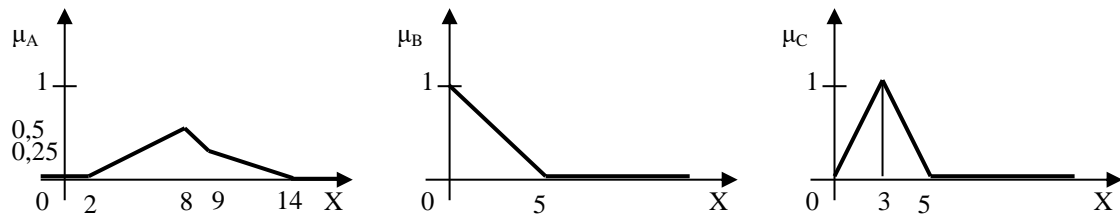


- 5) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \bar{B} \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

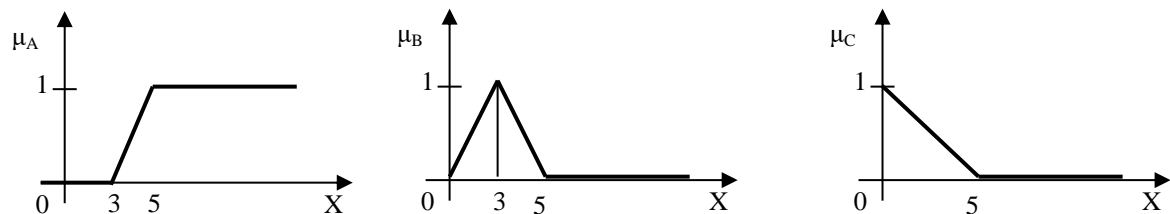


- 6) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cup \bar{B} \cap C$ и

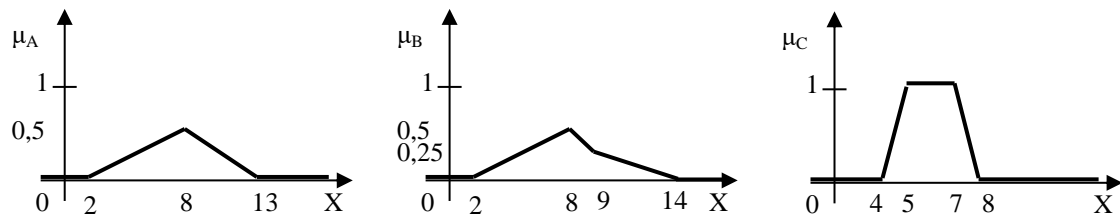
определить степень принадлежности одного элемента множеству D, используя метод ограничений.



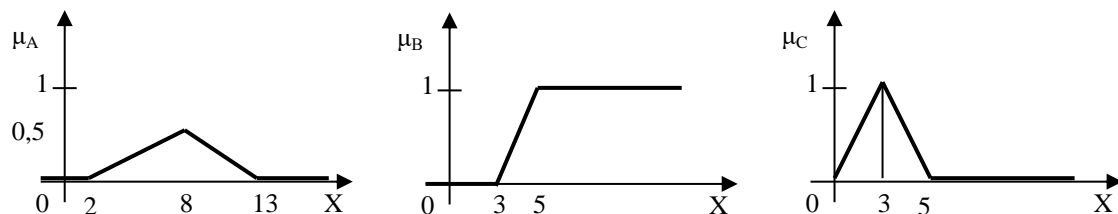
- 7) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cap B \cup \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.



- 8) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cap B \cup \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

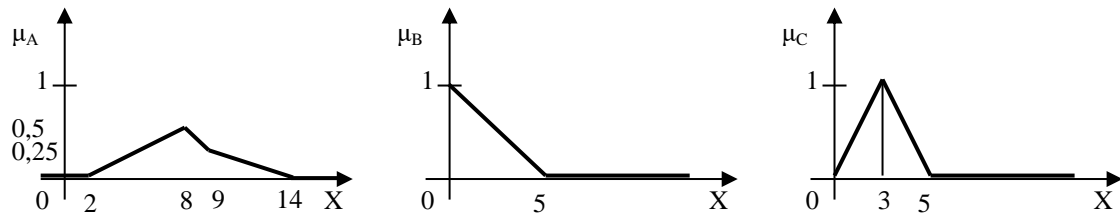


- 9) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = A \cap \bar{B} \cup \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

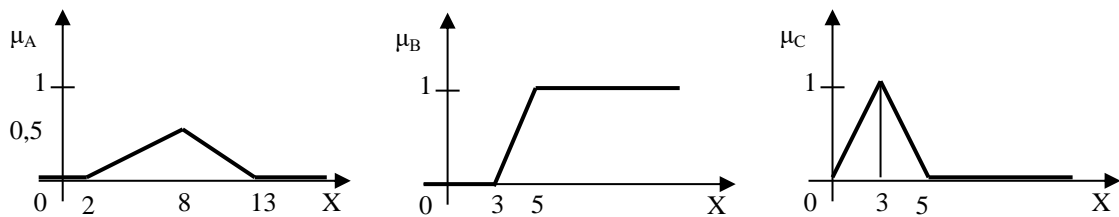


- 10) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \bar{A} \cup B \cap C$ и

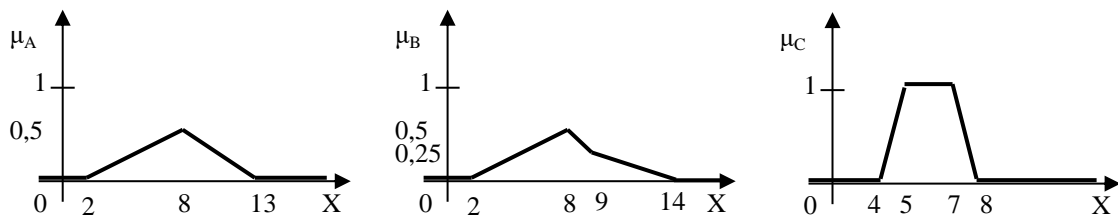
определить степень принадлежности одного элемента множеству D, используя максиминный способ.



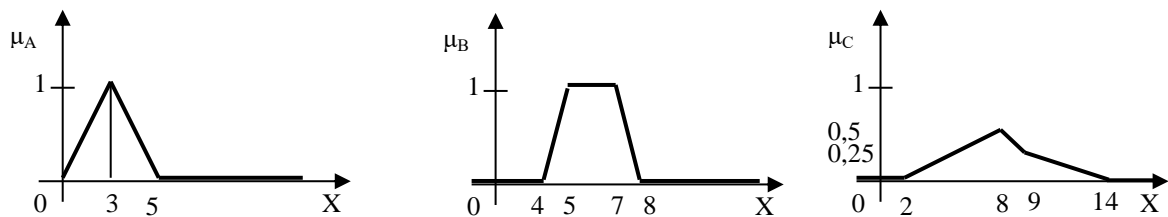
- 11) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \bar{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



- 12) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \bar{A} \cup B \cap C$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.

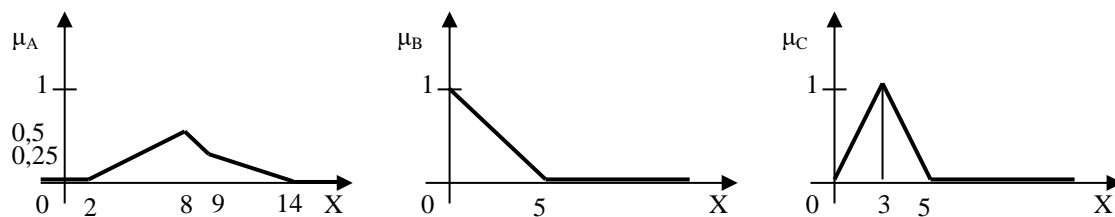


- 13) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = (\bar{A} \cup B) \cap \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

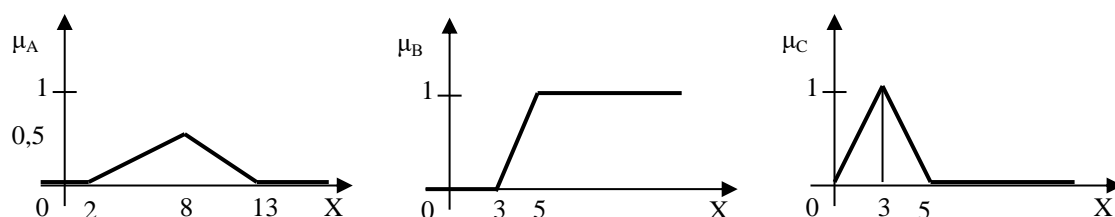


- 14) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = (\bar{A} \cup B) \cap \bar{C}$ и

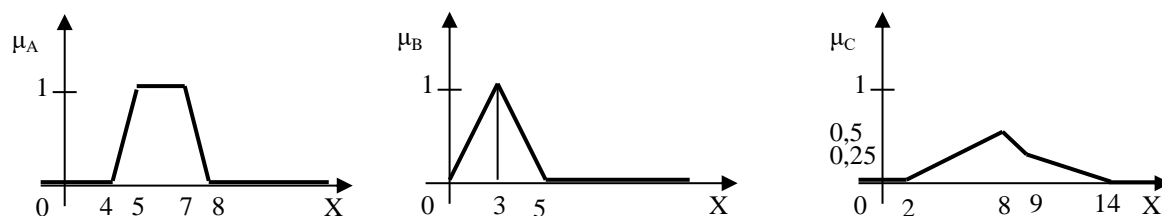
определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



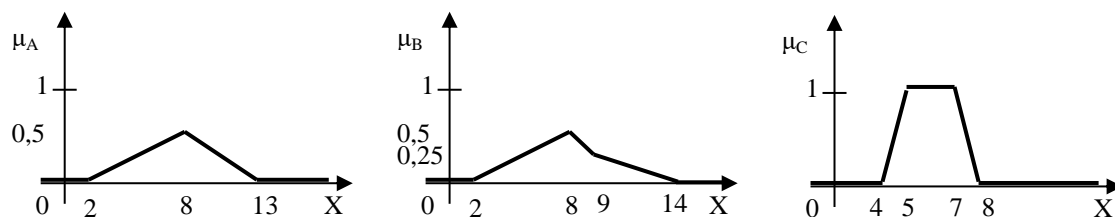
- 15) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = (\overline{A \cup B}) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



- 16) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя максиминный способ.

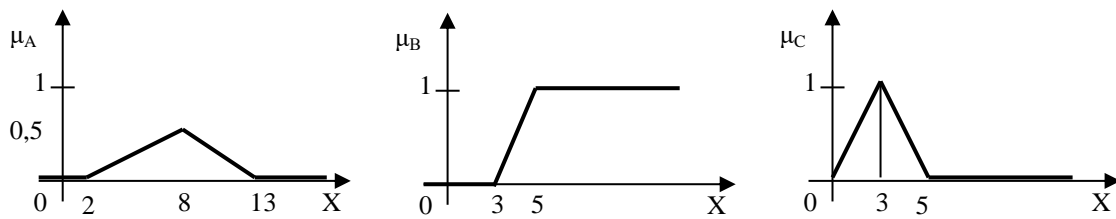


- 17) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.

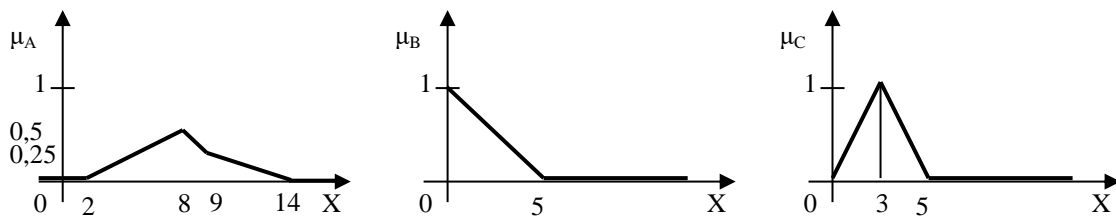


- 18) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \overline{A} \cap (C \cup B) \cap \overline{C}$ и

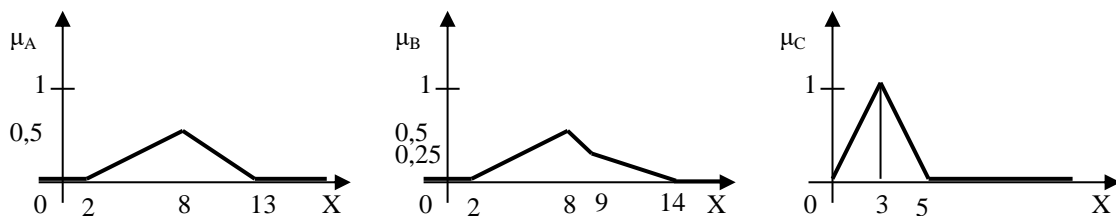
определить степень принадлежности одного элемента множеству D, используя метод ограничений.



- 19) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \bar{A} \cup \bar{B} \cap \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя алгебраический способ.



- 20) Дано 3 нечетких множества A, B, C (заданы их функции принадлежности). Построить функцию принадлежности нечеткого множества $D = \bar{A} \cup \bar{B} \cap \bar{C}$ и определить степень принадлежности одного элемента множеству D, используя метод ограничений.



Тема 6. Нейронные сети.

Основные вопросы темы

1. Понятие нейрона.
2. Модель математического нейрона.
3. Перцептрон Розенблатта. Правила Хебба.
4. Алгоритм обучения по дельта-правилу.
5. Обучение многослойной нейронной сети методом обратного распространения ошибки.
6. Классификация нейронных сетей.

7. Задачи, решаемые нейронными сетями.
8. Глубинное обучение.
9. Сверточные нейронные сети.
10. Рекуррентные нейронные сети.

Рекомендации по изучению темы

Вопрос 1 изложен в учебнике [1] на с. 50-53.

Вопрос 2 изложен в учебнике [1] на с. 51.

Вопрос 3 изложен в учебнике [1] на с. 52.

Вопрос 4 изложен в учебнике [1] на с. 55-56.

Вопрос 5 изложен в учебнике [1] на с. 57-58

Вопрос 6 изложен в учебнике [1] на с. 53-55.

Вопрос 7 изложен в учебнике [1] на с. 59-60.

Дополнительный материал по вопросам представлен в [4] главе 4 и в [7].

Вопросы для самоподготовки

Рекомендуется после изучения материалов лекций и специальной литературы подготовить ответы на вопросы:

1. Что такое нейрон?
2. Что такое искусственная нейронная сеть?
3. Какие нейронные сети относятся к глубинным?
4. Что собой представляет процесс обучения?
5. Какие существуют проблемы обучения?
6. Какие достоинства и недостатки у НС?
7. Чем отличается обучение с учителем от обучения без учителя?
8. Какие задачи решают искусственные НС?
9. По каким параметрам можно классифицирования искусственные НС?
10. Какие функции используют в качестве функций активации нейронов?
11. Как инициализируют синаптические веса НС?
12. Что из себя представляет пакетная нормализация?
13. Как выполняется операция свёртка в сверточной НС?
14. Как используется градиент в обучении НС?

Контрольные тесты

- 1) Искусственные нейронные сети— модели машинного обучения, использующие комбинации распределенных простых операций, зависящих**

от обучаемых параметров, для обработки входных данных. Какого вида ИНС не существует?

Выберите один или несколько ответов:

- a. соревновательные
- b. прямого распространения
- c. наивные
- d. реактивные
- e. рекуррентные

2) Что из ниже перечисленного относится к персептрону?

Выберите один или несколько ответов:

- a. нейронная сеть с обратными связями
- b. многослойная нейронная сеть
- c. однослойная нейронная сеть
- d. создан У. Маккалоком и В. Питтом
- e. создан Ф. Розенблаттом
- f. нейронная сеть прямого распространения

3) Какие из перечисленных сетей являются рекуррентными?

Выберите один ответ:

- a. нет правильного ответа
- b. персептрон
- c. сеть Хопфилда
- d. сеть радиальных базисных функций

4) Как называется задача, решаемая ИНС и направленная на предсказание значения той или иной непрерывной числовой величины для входных данных?

Выберите один ответ:

- a. Классификация
- b. Кластеризация
- c. Переобучение

- d. Регрессия

5) Какую нейронную сеть обучают с помощью дельта-правила?

Выберите один или несколько ответов:

- a. сеть Хопфилда
- b. нейронную сеть с обратными связями
- c. однослойную нейронную сеть
- d. нейронную сеть прямого распространения

6) Какую нейронную сеть обучают с помощью алгоритма обратного распространения ошибки?

Выберите один ответ:

- a. многослойную нейронную сеть с обратными связями
- b. нет правильного ответа
- c. многослойную нейронную сеть прямого распространения
- d. Однослойную нейронная сеть

7) Какие задачи не решают нейронные сети?

Выберите один или несколько ответов:

- a. управление
- b. аппроксимация
- c. классификация
- d. память, адресуемая по содержанию
- e. маршрутизация
- f. кодирование

8) Какую функцию не может решить однослойная нейронная сеть?

Выберите один ответ:

- a. логическое «не»
- b. логическое «исключающее или»
- c. логическое «или»

9) Какой из видов машинного обучения основывается на взаимодействии обучаемой системы со средой?

Выберите один ответ:

- a. Обучение с подкреплением
- b. Обучение без учителя
- c. Глубинное обучение
- d. Обучение с учителем

10) В какие игры нейросеть еще не научилась обыгрывать человека?

Выберите один ответ:

- a. Го
- b. Бридж
- c. Шахматы
- d. «Марио»

Задания для практических занятий и самостоятельной работы

Пример решения

Задача. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс ($k=1$) и пороговую функцию ($T=0,7$). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

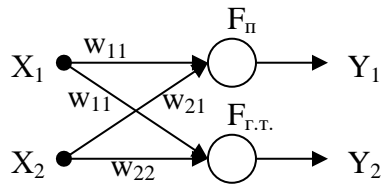
Описание процесса решения. Для обучения нейронной сети по Δ -правилу необходимо:

- 1) Графически отобразить структуру нейронной сети. Определить размерность матрицы синаптических весов.
- 2) Определить обучающую выборку, представив ее в табличном виде.
- 3) Выбрать входные данные, на которых будет рассматриваться итерация цикла обучения.
- 4) Следуя алгоритмы обучения по Δ -правилу, просчитать одну итерацию цикла и представить новые синаптические веса в матричном виде.

Решение.

- 1) По заданию нейронная сеть состоит из двух нейронов, значит, входов у однослойной нейронной сети будет 2 и выходов 2, а синаптических весов 4.

Первый нейрон имеет пороговую функцию активации, второй – гиперболический тангенс.



- 2) По заданию нейронная сеть бинарная, поэтому на ее входы могут подаваться только нули и единицы, так как входов 2, то возможных комбинаций входных значений будет 4 (обучающая выборка будет состоять из 4 векторов). Выход первого нейрона согласно заданию соответствует оператору эквивалентности, а второго – дизъюнкции. Поэтому таблица с обучающей выборкой будет выглядеть следующим образом:

X1	X2	D1	D2
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	1

- 3) Пусть в качестве вектора обучения будет рассматриваться 3-я строка таблицы.
 4) Следуя алгоритмы обучения по Δ -правилу выполним 6 шагов

1 шаг: зададим матрицу весов случайным образом из интервала $[0,1]$:

Wij(1)	1	2
1	0.7	1
2	0.5	0.2

2 шаг: вектор $X = \{1, 0\}$, вектор $D = \{0, 1\}$.

3 шаг: вычисление выходных значений нейронной сети (вектор Y).

$$T = 0.7;$$

$$S_1 = x_1 \cdot w_{11} + x_2 \cdot w_{21} = 1 \cdot 0.7 + 0 \cdot 0.5 = 0.7;$$

$$Y_1 = \begin{cases} 1, & \text{при } S_1 \geq T \\ 0, & \text{при } S_1 < T \end{cases} = \begin{cases} 1, & \text{при } 0.7 \geq 0.7 \\ 0, & \text{при } 0.7 < 0.7 \end{cases} = 1.$$

$$k = 1,$$

$$S_2 = x_1 \cdot w_{12} + x_2 \cdot w_{22} = 1 \cdot 0.9 + 0 \cdot 0.2 = 0.9,$$

$$Y_2 = th\left(\frac{S_2}{k}\right) = \frac{e^{0.9} + e^{-0.9}}{e^{0.9} - e^{-0.9}} \approx 1.39.$$

4 шаг:

$$\varepsilon_1 = (d_1 - y_1) = (0 - 1) = -1,$$

$$\varepsilon_2 = (d_2 - y_2) = (1 - 1.39) = -0.39.$$

5 шаг: задаем η - коэффициент обучения от 0 до 1 и изменяем веса:

$$\eta = 0.8,$$

$$w_{11}(2) = w_{11}(1) - 0.8 \cdot \varepsilon_1 \cdot x_1 = 0.7 - 0.8 \cdot (-1) \cdot 1 = 1.5,$$

$$w_{21}(2) = w_{21}(1) - 0.8 \cdot \varepsilon_1 \cdot x_2 = 0.5 - 0.8 \cdot (-1) \cdot 0 = 0.5,$$

$$\theta_1(2) = \theta_1(1) - 0.8 \cdot \varepsilon_1 = 0.7 - 0.8 \cdot (-1) = 1.5,$$

$$w_{12}(2) = w_{12}(1) - 0.8 \cdot \varepsilon_2 \cdot x_1 = 0.9 - 0.8 \cdot (-0.39) \cdot 1 = 1.212,$$

$$w_{22}(2) = w_{22}(1) - 0.8 \cdot \varepsilon_2 \cdot x_2 = 0.2 - 0.8 \cdot (-0.39) \cdot 0 = 0.2.$$

Wij(2)	1	2
1	1.5	1.212
2	0.5	0.2

6 шаг: вычислим среднеквадратичную ошибку (можно выбрать другие методы оценки ошибки)

$$\varepsilon = \sum_{i=1}^H (d_i - y_i)^2 = \sum_{i=1}^2 \varepsilon_i^2 = \varepsilon_1^2 + \varepsilon_2^2 = (-1)^2 + (-0.39)^2 = 1.1521.$$

H - количество нейронов.

Так как мы рассматриваем одну итерацию цикла обучения в любом случае выходим из цикла.

Варианты задач

1. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей пороговую функцию активации ($T=0,7$). В качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
2. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей линейную функцию активации ($k=0,6$). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
3. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей сигмоидальную функцию активации ($k=1$). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

4. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 2 нейронов и имеющей функцию активации гиперболический тангенс ($k=1$). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
5. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс ($k=2$) и пороговую функцию ($T=0,5$). В качестве обучающей выборки использовать таблицу истинности для операций эквивалентности и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
6. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: сигмоидальную ($k=1$) и линейную ($k=0,6$). В качестве обучающей выборки использовать таблицу истинности для операций импликации и конъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
7. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: линейную ($k=0,7$) и пороговую ($T=0,75$). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
8. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: пороговую ($T=0,8$) и сигмоидальную ($k=1$). В качестве обучающей выборки использовать таблицу истинности для операций конъюнкции и импликации (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
9. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс ($k=2$) и линейную ($k=0,8$). В качестве обучающей выборки использовать таблицу истинности для операций дизъюнкции

и эквивалентности (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.

10. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной неоднородной нейронной сети, состоящей из 2 нейронов и имеющей функции активации: гиперболический тангенс ($k=2$) и сигмоидальную ($k=0,9$). В качестве обучающей выборки использовать таблицу истинности для операций импликации и дизъюнкции (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
11. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей функцию активации гиперболический тангенс ($k=3$). В качестве обучающей выборки использовать таблицу истинности для $X_1 \& X_2 \rightarrow X_3$, $X_1 \& X_2$ и $X_2 \rightarrow X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
12. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей сигмоидальную функцию активации ($k=1$). В качестве обучающей выборки использовать таблицу истинности для $X_1 \rightarrow X_2 \& X_3$, $X_1 \& X_2$ и $X_1 \& X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
13. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей линейную функцию активации ($k=0,9$). В качестве обучающей выборки использовать таблицу истинности для $X_3 \rightarrow X_1 \& X_2$, $X_2 \& X_3$, $X_2 \rightarrow X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
14. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной бинарной однородной нейронной сети, состоящей из 3 нейронов и имеющей пороговую функцию активации ($T=0,4$). В качестве обучающей выборки использовать таблицу истинности для $(X_2 \rightarrow X_1) \& X_3$ (не использовать первую строчку таблицы). Синаптические веса задать случайным образом.
15. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей линейную функцию активации ($k=0,9$). Синаптические веса и обучающую выборку задать случайным образом (не нули).
16. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей сигмоидальную

функцию активации ($k=0,8$). Синаптические веса и обучающую выборку задать случайным образом (не нули).

17. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей пороговую функцию активации ($T=0,8$). Синаптические веса и обучающую выборку задать случайным образом (не нули).
18. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой однородной нейронной сети, состоящей из 3 нейронов и имеющей функцию активации – гиперболический тангенс ($k=1$). Синаптические веса и обучающую выборку задать случайным образом (не нули).
19. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой неоднородной нейронной сети, состоящей из 3 нейронов и имеющей функции активации: сигмоидальную ($k=1$), линейную ($k=0,8$) и пороговую ($T=0,5$). Синаптические веса и обучающую выборку задать случайным образом (не нули).
20. Просчитать одну итерацию цикла обучения по Δ -правилу однослойной аналоговой неоднородной нейронной сети, состоящей из 3 нейронов и имеющей функции активации: гиперболический тангенс ($k=1$), сигмоидальную ($k=0,8$) и пороговую ($T=0,6$). Синаптические веса и обучающую выборку задать случайным образом (не нули).

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Порядок выполнения лабораторных работ может быть произвольным и определяется уровнем освоения компетенций обучающегося.

Тема 3. Онтологии и Semantic Web

Задание лабораторной работы

Цель работы: получение практических навыков построения онтологий.

Задание: Используя программу Protege:

- 1) создать онтологию согласно полученному варианту, онтология должна содержать:
 - иерархию классов (не менее 15);
 - назначенные классам простые свойства Data Properties (не менее 10);
 - назначенные классам свойства-отношения Object Properties (не менее 5) с указанием вида связи между индивидами (функциональная, симметричная и т.д.);
 - индивиды Individuals (не менее 10), с заполненными значениями свойств унаследованного класса;
 - аксиомы, наложенные на свойства и классы в Equivalent to и др. (не менее 5);
 - правила (не менее 5);
- 2) используя плагин OntoGraf (вкладка Window/Tabs/OntoGraf), получить визуальное отображение онтологии в виде графа;
- 3) онтология должна охватывать всю предметную область (требование полноты), и быть достаточно детализированной;
- 4) используя плагин SQWRLTab, построить запросы к онтологии (не менее 3);
- 5) сохранить онтологию;
- 6) открыть созданную онтологию и сохранить во второй файл, затем, используя плагин SWRLTab, построить прогнозируемые аксиомы и дополнить ими онтологию. Провести сравнительный анализ двух онтологий, проверить правильность полученных автоматически элементов онтологии.

Отчет по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание
2. Описание онтологий:
 - a. структуры;

- b. описание классов;
 - c. свойства;
 - d. индивиды;
3. Графическое отображение онтологий.

Методические указания по выполнению лабораторной работы

Онтология – формальное явное описание понятий в рассматриваемой предметной области (**классов** (иногда их называют **понятиями**)), свойств каждого понятия, описывающих различные свойства и атрибуты понятия (**слотов** (иногда их называют **ролями** или **свойствами**)), и ограничений, наложенных на слоты (**фацетов** (иногда их называют **ограничениями ролей**)).

Элементы онтологий:

- понятия (классы) – концептуальные сущности;
- экземпляры (индивиды) – конкретные представители понятий;
- отношения (предикаты) – связи между понятиями или экземплярами (таксономические, родственные, производственные), а также свойства сущности (цвет, длина, вкус);
- аксиомы – очевидные утверждения, связывающие понятия, экземпляры и отношения

Онтология вместе с набором индивидуальных **экземпляров** классов образует **базу знаний**. В действительности, трудно определить, где кончается онтология и где начинается база знаний.

Классы описывают понятия предметной области. Например, класс вин представляет все вина. Конкретные вина – экземпляры этого класса. Вино в конкретном бокале – это экземпляр класса вин. Класс может иметь подклассы, которые представляют более конкретные понятия, чем надкласс. Например, мы можем разделить класс всех вин на красные, белые и розовые вина. В качестве альтернативы мы можем разделить класс всех вин на игристые и не игристые вина.

Слоты описывают свойства классов и экземпляров: вино крепкое, оно производится на винном заводе Абрау-Дюрсо. У нас есть два слота, которые описывают вино в этом примере: слот крепость со значением «крепкое» и слот производитель со значением «винный завод Абрау-Дюрсо». Мы можем сказать, что на уровне класса у экземпляров класса Вино есть слоты, которые описывают вкус, крепость, уровень сахара, производителя вина и т.д.

Все экземпляры класса Вино и его подкласс Красное вино имеют слот производитель, значение которого является экземпляром класса Винный завод. Все экземпляры класса Винный завод имеют слот производит, относящийся ко всем винам (экземплярам класса Вино и его подклассов), которые производятся на этом заводе.

Правила построения онтологий.

- 1) Не существует единственного правильного способа моделирования предметной области – всегда существуют жизнеспособные альтернативы. Лучшее решение почти всегда зависит от предполагаемого приложения и ожидаемых расширений.
- 2) Разработка онтологии – это обязательно итеративный процесс.
- 3) Понятия в онтологии должны быть близки к объектам (физическим или логическим) и отношениям в интересующей вас предметной области. Скорее всего, это существительные (объекты) или глаголы (отношения) в предложениях, которые описывают вашу предметную область.

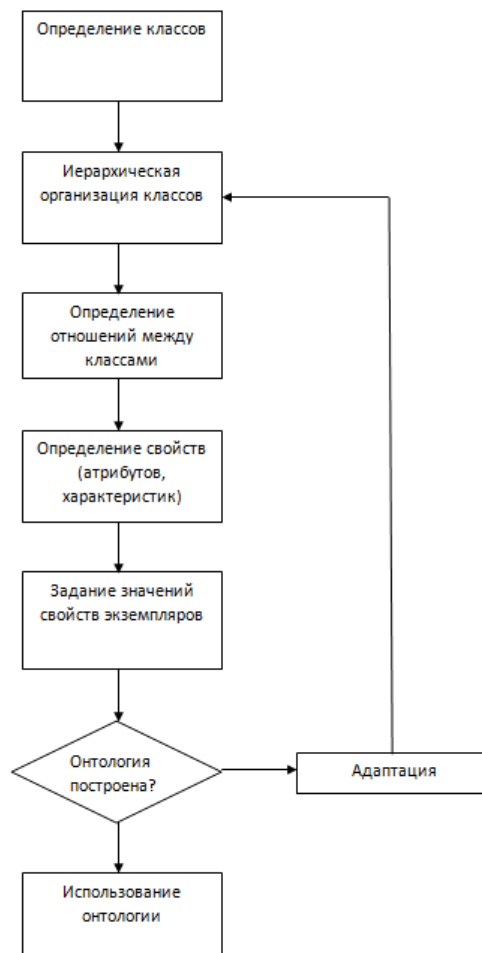


Рисунок 1 - Схема процесса создания онтологии

Аксиомы построения онтологий:

- *Аксиома 1.* Правильного способа моделирования предметной области нет, поскольку всегда есть хорошие альтернативы. Оптимальное решение зависит от предполагаемого результата.
- *Аксиома 2.* Разработка и описание онтологий всегда носит постоянный итеративный характер.
- *Аксиома 3.* Понятия в онтологии всегда напрямую связаны с физическими или логическими объектами, а также отношениями в конкретной предметной области.

Характеристики свойств объектов

Функциональные свойства (Functional) – если свойство является функциональным, то для данного индивида (экземпляра) может существовать не более одного индивида, который имеет отношение к первому индивиду через это свойство.

Обратные функциональные свойства (Inverse functional) – если свойство является обратным функциональному свойству, то это значит, что свойство является обратным функциональным.

Транзитивные свойства (Transitive) – если свойство транзитивное и свойство связывает индивида а и индивида b, а также индивида b связывает с индивидом с, то мы можем вывести, что индивид а связан с индивидом с через это свойство.

Симметричные свойства (Symmetric) – если свойство р симметричное, и свойство связывает индивида а с индивидом b, то индивид b связан также с индивидом а через свойство р.

Асимметричные свойства (Asymmetric) – если свойство р асимметричное, и свойство связывает индивида а с индивидом b, то индивид b не может быть связан с индивидом а через свойство р.

Рефлексивные свойства (Reflexive) – свойство р называется рефлексивным, когда индивид а должен быть связан с собой.

Иррефлексивные свойства (Irreflexive) – если свойство р иррефлексивное, то оно может быть охарактеризовано как свойство, которое связывает индивида а с индивидом b, где индивид а и индивид b обязательно разные.

OWL (Web Ontology Language) вобрал в себя всё лучше из DAML+OIL и является дальнейшим развитием RDF/RDFS, функционально-ориентированный стиль описания онтологий.

Список команд для построения аксиом (можно применять в Equivalent to)

Команда	Пример	Значение
some	hasPet some Dog	Things that have a pet that is a Dog

value	hasPet value Tibbs	Things that have a pet that is Tibbs.
only	hasPet only Cat	Things that have pets that are only Cats.
min	hasPet min 3 Cat	Things that have at least three pets that are Cats.
max	hasPet max 5 Dog	Things that have at most five pets that are Dogs.
exactly	hasPet exactly 2 GoldFish	Things that have exactly 2 GoldFish as pets.
and	Person and (hasPet some Cat)	People that have a pet that's a Cat.
or	(hasPet some Cat) or (hasPet some Dog)	Things that have a pet that's a Cat or have a pet that's a Dog
not	not (hasPet some Dog)	Things that do not have a pet that's a dog.

SPARQL(SPARQL Protocol and RDF Query Language):

- язык запросов к RDF данным и протокол связи с RDF-хранилищами,
- аналог языка SQL для реляционных баз данных,
- SPARQL-запрос представляет собой набор графовых шаблонов, с которыми сравниваются имеющиеся RDF-данные.

Общая схема SPARQL-запроса SELECT

► PREFIX

- # префиксные объявления - служат для указания сокращений универсальных идентификаторов ресурса (URI),
- # используемых в запросе.

► FROM ...

- # источники запроса - определяют какие RDF-графы запрашиваются.

► SELECT ...

- # состав результата - определяет возвращаемые элементы данных.

► WHERE {...}

- # шаблон запроса - определяет, что запрашивать из набора данных.

► ORDER BY ...

- # модификаторы запроса - ограничивают, упорядочивают, преобразуют результаты запроса

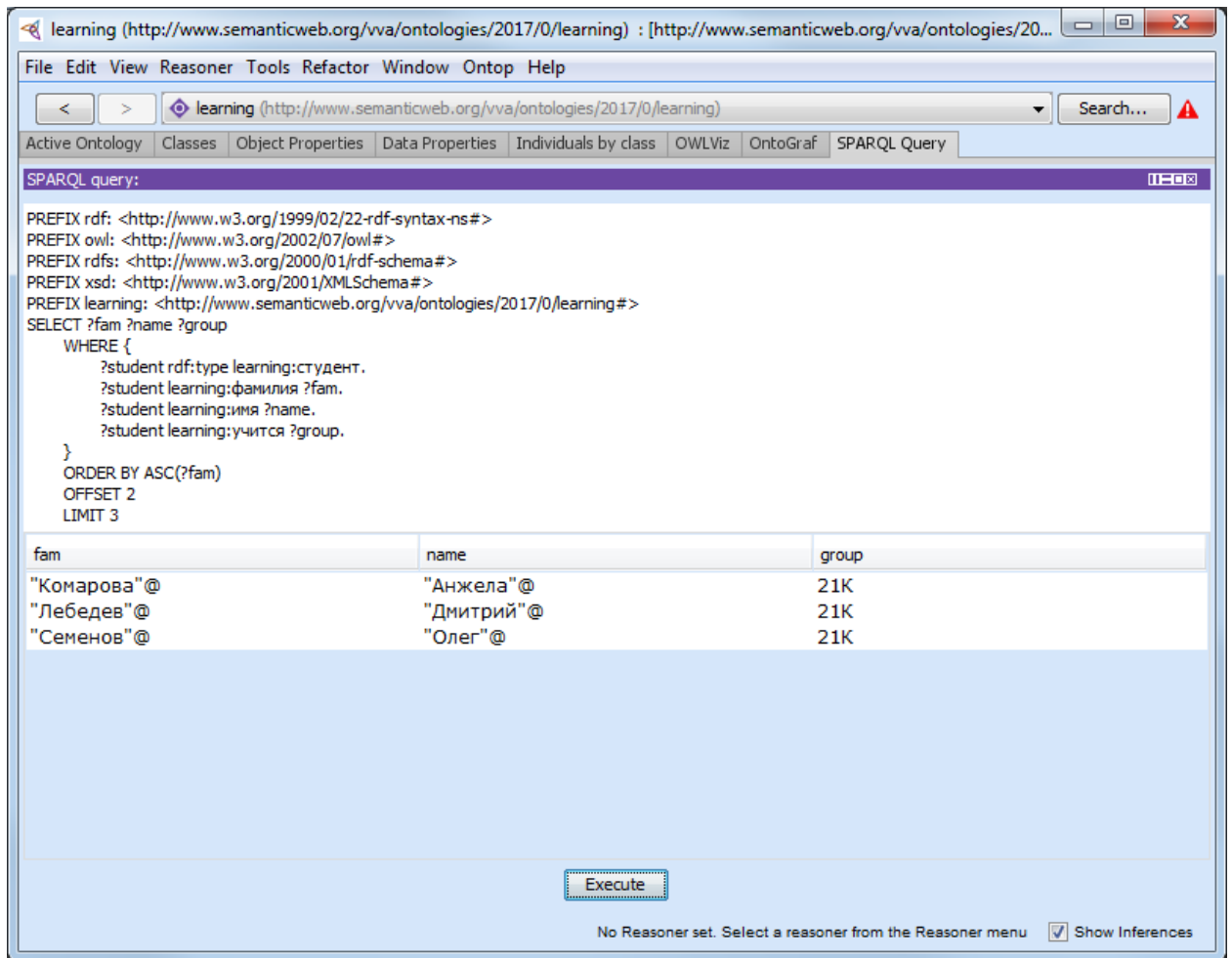


Рисунок 2 - Пример запроса к онтологии

SWRL (Semantic Web Rule Language)

- это язык, созданный для организации логического вывода на OWL(Web Ontology Language) онтологии.

- Работа осуществляется с хорновскими дизъюнктами. Дизъюнктом Хорна называют выражение

$$a_1 \wedge a_2 \wedge \dots \wedge a_N \rightarrow b$$

Ограничения SWRL

- Отрицание, Дизъюнкция
- Неразрешимость
- Не очень хорошо понятен: для него не существует родных механизмов суждений (reasoners), не размечены четко отношения с другими формализмами.

Типы утверждений SWRL

1. **Утверждения о принадлежности к классу.** Записывается в следующей форме:

Человек (Петр). Аргументом является индивидуальный объект. Минимальное SWRL-правило может выглядеть так:

Мужчина (?x) -> Человек (?x)

Это правило утверждает, что любой экземпляр класса Мужчина является и экземпляром класса Человек. Впрочем, столь простой вывод можно получить и средствами OWL, без использования SWRL. Как мы видим, переменные в SWRL, как и в SPARQL, предваряются знаком вопроса; знак -> служит для разделения условий и вывода.

2. **Утверждения о существовании связи.** Примером является уже приводившееся выражение являетсяСыном (Петр, ?x) – заметим, что оно будет иметь практический смысл в правиле только в том случае, если хотя бы один аргумент будет переменной.

3. **Утверждения о значении свойства-литерала.** При помощи способа, аналогичного приведенному в предыдущем пункте, можно получать в переменные значения свойств-литералов, и затем сравнивать их с различными выражениями при помощи встроенных функций. Пример: имеетВозраст (?x , ?y) \wedge swrlb:greaterThanOrEqual (?y, 16).

4. **Условие раздельности двух индивидуальных объектов:** differentFrom (?x, ?y)

5. **Условие совпадения двух объектов:** sameAs (?x, ?y). Условие будет истинно, если переменные ?x и ?y оказались равны одному и тому же объекту.

6. **Условие о принадлежности значения переменной определенному типу данных.**

Например: xsd:int (?x)

7. Так называемые **встроенные условия (built-ins)** : swrlb:greaterThanOrEqual.

Схема утверждений SWRL:

- ▶ Унарные атомы:
 - ▶ C(arg1), где C – произвольное выражение класса OWL DL
 - ▶ D(dataArg1) где D – тип данных URI или диапазон перечисленных значений
- ▶ Бинарные атомы:
 - ▶ P(arg1, arg2) где P – свойство объекта
 - ▶ Q(arg1, dataArg1) где Q -свойство типа данных
 - ▶ arg1 = arg1 равенство, или “sameAs”
 - ▶ arg1 \neq arg2 неравенство, или “differentFrom”
- ▶ Аргументы
- ▶ arg1 | arg2 - либо индивиды, обозначающие URI, либо переменные, задающие диапазон индивидов

- ▶ dataArg1 - либо литералы данных, либо переменные, задающие диапазон значений данных

Семантический механизм рассуждений (англ. *semantic reasoner*), **семантическая машина формирования рассуждений** (англ. *reasoning engine*) или **движок правил** (англ. *rules engine*) — это часть программного обеспечения, способная вывести логические умозаключения из набора адекватно формализованных базовых знаний или аксиом.

Понятие семантического механизма рассуждений обобщает понятие машины вывода, предоставляя более богатый набор механизмов для работы.

Правила вывода обычно определяются с помощью языка онтологий и часто языков описательной логики.

Многие семантические механизмы рассуждений используют логику первого порядка для выполнения рассуждений; вывод обычно происходит путём прямой и обратной цепочек рассуждений.

Варианты для выполнения лабораторной работы

1. Информационные системы
2. Интеллектуальные информационные системы
3. Компьютерные сети
4. Языки программирования
5. Модели представления знаний
6. Классификация нейронных сетей
7. Классификация экспертных систем
8. Классификация систем поддержки принятия решений
9. Электронно-вычислительные машины (компьютеры)
10. Направления искусственного интеллекта
11. Операционные системы
12. Классификация наук
13. Направления исследований в информатике
14. Протоколы Интернет
15. Свободный вариант, студент сам предлагает предметную область (связанна с информатикой) и согласовывает с преподавателем.

Тема 4. Эволюционное моделирование

Задание лабораторной работы

Цель работы: получение практических навыков использования генетических алгоритмов на языке Python с использованием библиотеки DEAP.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку DEAP, NumPy, Matplotlib и др. реализовать генетический алгоритм согласно варианту.

Отчёт по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание, вариант.
2. Описание построенного генетического алгоритма и его операторов.
3. Протокол прогона ГА: особи популяций, лучшие особи, приспособленность особей, максимальное значение приспособленности, минимальное значение приспособленности.
4. График изменения параметров ГА.
5. Выполнить 3 прогона с разными параметрами генетического алгоритма, сравнить результаты и определить лучший вариант параметров (который быстрее привёл к результату / дал лучший результат).
6. Код.

Методические указания по выполнению лабораторной работы

Элементы библиотеки DEAP для генетического алгоритма

Библиотека DEAP (Distributed Evolutionary Algorithms in Python) содержит распределённые эволюционные алгоритмы в Python.

Для реализации этих эволюционных алгоритмов, требуется использовать пакеты библиотеки, среди которых можно выделить ядро, реализующее базовые классы и инструменты для алгоритмов, и пакеты, реализующие различные эволюционные методы (Рисунок 3, Таблица 1).

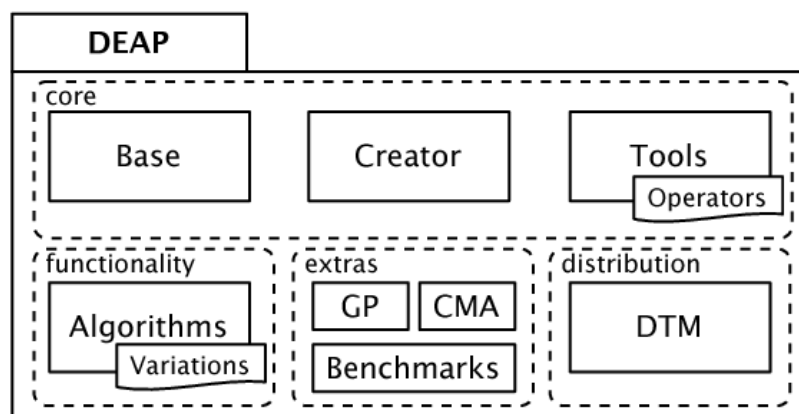


Рисунок 3 – Схема классов библиотеки¹

Таблица 1 - Модули библиотеки для реализации ГА

Модуль	Описание
deap.creator	Содержит класс <code>deap.creator.create</code> , необходим для создания новых классов с именем <code>name</code> , наследующий классы из пакета <code>base</code> . Новый класс может иметь разные атрибуты.
deap.base	Пакет содержит базовые классы. <ol style="list-style-type: none"> 1) <code>deap.base.Toolbox</code> - набор инструментов для эволюционных методов (на основе классов пакета <code>tools</code>), 2) <code>deap.base.Fitness</code> - пригодность / приспособленность (показатель качества решения) особи. 3) <code>class deap.base.Tree</code> (для версии DEAP 0.8.2) Базовый класс N-арного дерева. Дерево инициализируется из содержимого списка. Первый элемент списка является корнем дерева, затем следующие элементы являются узлами. Каждый узел может быть либо списком, либо отдельным элементом. В случае списка это рассматривается как поддереву, иначе лист.
deap.tools	Модуль инструментов содержит операторы для эволюционных алгоритмов. Набор операторов, которые он содержит, доступен на панели инструментов, также модуль содержит служебные инструменты для фиксации данных о функционировании алгоритмов.
deap.algorithms	Модуль алгоритмов содержит основные реализации эволюционных методов, при этом используются операторы, зарегистрированные в соответствующем объекте <code>Toolbox</code> . Обычно используются следующие ключевые слова: <code>mate ()</code> для кроссовера, <code>mutate ()</code> для мутации, <code>select ()</code>

¹ <https://www.semanticscholar.org/paper/DEAP%3A-a-python-framework-for-evolutionary-Rainville-Fortin/7627ef0d9975dc50f81f6894f976336c31bb6a38>

для выбора.

Реализация генетического алгоритма

Определение приспособленности особей и вида особи (индивида)

Необходимо определить класс для приспособленности и особи (индивида), для этого нужно использовать пакет `deap.creator` и класс `Creator`:

```
deap.creator.create(name, base[, attribute[, ...]])
```

Первый параметр *name* задаёт имя класса. Базой (родителем) для пользовательского класса приспособленности является `base.Fitness` (можно определить другой класс, если он реализован программистом).

При использовании алгоритмов `deap` в качестве третьего атрибута необходимо задать параметр *weights*. Нужно определить тип задачи оптимизации (максимум / минимум и однокритериальная / многокритериальная). В зависимости от этого определяется параметр *weights* (Таблица 2). Веса могут также использоваться, чтобы варьировать важность критериев оптимизации друг относительно друга (чем больше вес, тем важнее критерий), т.е. для задание многоцелевой функции оптимизации. Это означает, что весами могут быть любые действительные числа, и только знак используется для определения того, выполняется ли максимизация или минимизация.

Таблица 2 – Примеры определения параметра *weights*

Тип задачи оптимизации	Значения параметра <i>weights</i>
Однокритериальная на минимум	<code>weights=(-1.0,)</code>
Однокритериальная на максимум	<code>weights=(1.0,)</code>
Многокритериальная оптимизация (2 критерия)	<code>weights=(-1.0, 1.0)</code>

Пример:

- 1) Однокритериальная оптимизация на максимум

```
from deap import creator
creator.create("F", base.Fitness, weights=(1.0,))
```

- 2) Однокритериальная минимизация с именем `FitnessMin`.

```
from deap import creator
creator.create("FitnessMin", base.Fitness, weights = (- 1.0,))
```

- 3) Этот код создаёт соответствие, которое минимизирует первую цель и максимизирует вторую.

```
from deap import creator
creator.create ("FitnessMulti", base.Fitness, weights = (- 1.0, 1.0))
```

Для определения вида особи используется этот же класс, созданный ранее класс приспособленности становится значением параметра *fitness*.

В качестве особи может использоваться массив, причём его элементы могут быть разного типа. Для определения типа массива можно использовать параметр *typecode* (Таблица 3).

Таблица 3 – Коды типов

Код типа	Тип в python	Минимальный размер в байтах
'b'	int	1
'B'	int	1
'h'	int	2
'H'	int	2
'i'	int	2
'I'	int	2
'l'	int	4
'L'	int	4
'q'	int	8
'Q'	int	8
'f'	float	4
'd'	float	8

Пример:

1) Особь - простой список, содержащий вещественные числа
`creator.create("Individual_1", list, fitness=creator.F)`

2) Особь – массив из пакета `array`
`import array`
`creator.create("Individual_2", array.array, typecode='b', fitness=creator.FitnessMin)`

3) Особь – массив из пакета `numpy`
`import numpy`
`creator.create("Individual_3", numpy.ndarray, fitness=creator.FitnessMulti)`

Регистрация операторов генетического алгоритма

Для определения параметров и операторов ГА используется класс `base.Toolbox()`, создаётся объект этого класса с помощью метода `base.Toolbox()`. В нем регистрируются операторы ГА:

`register(alias, method[, argument[, ...]])`.

Методу даётся псевдоним и указываются аргументы. Основные операторы ГА это:

- Инициализация (генерация особи)
- Формирование популяции
- Отбор родителей
- Скрещивание или кроссовер
- Мутация
- Миграция (для островной модели)

Для каждого оператора существуют разные методы и в модуле представлены варианты их реализаций (Таблица 4), кроме того программист может определить свои методы и задать их в качестве параметра функции `register`.

Таблица 4 – Список методов для реализации операторов ГА

Инициализация	Отбор	Скрещивание	Мутация
<code>initRepeat()</code>	<code>selTournament()</code>	<code>cxOnePoint()</code>	<code>mutGaussian()</code>
<code>initIterate()</code>	<code>selRoulette()</code>	<code>cxTwoPoint()</code>	<code>mutShuffleIndexes()</code>
<code>initCycle()</code>	<code>selNSGA2()</code>	<code>cxUniform()</code>	<code>mutFlipBit()</code>
	<code>selNSGA3()</code>	<code>cxPartiallyMatched()</code>	<code>mutPolynomialBounded()</code>
	<code>selSPEA2()</code>	<code>cxUniformPartiallyMatched()</code>	<code>mutUniformInt()</code>
	<code>selRandom()</code>	<code>cxOrdered()</code>	<code>mutESLogNormal()</code>
	<code>selBest()</code>	<code>cxBlend()</code>	
	<code>selWorst()</code>	<code>cxESBlend()</code>	
	<code>selTournamentDCD()</code>	<code>cxESTwoPoint()</code>	
	<code>selDoubleTournament()</code>	<code>cxSimulatedBinary()</code>	
	<code>selStochasticUniversalSampling()</code>	<code>cxSimulatedBinaryBounded()</code>	
	<code>selLexicase()</code>	<code>cxMessyOnePoint()</code>	
	<code>selEpsilonLexicase()</code>		Миграция
	<code>selAutomaticEpsilonLexicase()</code>		<code>migRing()</code>

Инициализация

Если значения генов выбираются случайно и могут повторяться, то можно использовать `initRepeat()`, т.е. случайная перестановка допустимых значений генов.

Если необходимо создать особь, значение генов в которой не повторяются, то можно использовать `initIterate()`.

Если необходимо генерировать хромосомы с заданной структурой (например 2 числа, одно целое другое вещественное, или 2 символа: первый цифра, второй латинская буква,

третий – русская) и известно сколько хромосом должно быть в особи (сколько раз повторяться сочетания), то можно использовать функцию *initCycle()*.

Пример:

- 1) Инициализация значениями 0 или 1, причём длина особи = 10.

```
import random
from deap import tools
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual_1,
toolbox.attr_bool, 10)
```

- 2) Особи из 5 ген, которые принимают значения от 0 до 10.

```
import random
from deap import tools
...
toolbox.register("indices", random.sample, range(10), 5)
toolbox.register("individual", tools.initIterate, creator.Individual_1,
toolbox.indices)
```

- 3) Циклическая инициализация в особи 4 пары чисел, в паре первое число целое в промежутке от 5 до 10, второе вещественное, в интервала от 0.1 до 0.7

```
import random
from deap import tools
toolbox = base.Toolbox()
toolbox.register("attr_int", random.randint, 5, 10)
toolbox.register("attrflt", random.uniform, 0.1, 0.7)
toolbox.register("individual", tools.initCycle, creator.Individual_1,
(toolbox.attr_int, toolbox.attrflt), n=4)
```

Отбор

Существует много стратегий отбора, в модуле *tools* реализовано 14 (Таблица 4), рассмотрим 5 из них (Таблица 5), для которых нужен следующий набор параметров:

- *individuals* – особи;
- *k* – количество отбираемых особей;
- *fit_attr* – атрибут, по которому осуществляется отбор;
- *tournament_size* - количество участников тура (для турнирного отбора).

Таблица 5 – Пример операторов отбора для ГА

Функция	Параметры	Вид	Описание
<code>deap.tools.selTournament</code>	$(individuals, k, \text{tournsize}, \text{fit_attr}='fitness')$	Турнирный отбор	из популяции, содержащей m особей, выбирается случайным образом t особей и выбирается наиболее приспособленная (между выбранными особями проводится турнир), эта операция повторяется m раз
<code>deap.tools.selRoulette</code>	$(individuals, k, \text{fit_attr}='fitness')$	Отбор рулеткой	вид пропорционального отбора, когда особи отбираются с помощью n «запусков» рулетки (колесо рулетки содержит по одному сектору для каждого члена популяции, размер i -ого сектора пропорционален соответствующей величине $P_s(i)$)
<code>deap.tools.selRandom</code>	$(individuals, k)$	Выбор случайных k особей	Случайная отбор
<code>deap.tools.selBest</code>	$(individuals, k, \text{fit_attr}='fitness')$	Выбор лучших k особей	Отбор усечением
<code>deap.tools.selWorst</code>	$(individuals, k, \text{fit_attr}='fitness')$	Выбор худших k особей	Отбор усечением наоборот

Подробнее см. <https://deap.readthedocs.io/en/master/api/tools.html>.

Пример:

```
toolbox.register("select", tools.selTournament, tournsize=3)
```

Скращивание

В модуле `tools` представлена реализация 12 видов операторов кроссовера (Таблица 4).

Рассмотрим подробнее 4 из них (

Таблица б), для них требуются переменные:

- *ind1* – первый родитель
- *ind2* – второй родитель
- *indpb* – вероятность (для равномерного кроссовера)

Таблица 6 – Пример операторов кроссовера

Функция	Параметры	Тип кроссовера	Описание	Пример
cxOnePoint()	(<i>ind1, ind2</i>)	Одноточечный	выбирается одна точка разрыва и родительские хромосомы обмениваются одной из получившихся частей	Родитель 1: 1001011 01001 Родитель 2: 0100011 00111 Потомок 1: 1001011 00111 Потомок 2: 0100011 01001
cxTwoPoint()	(<i>ind1, ind2</i>)	Двухточечный	выбираются две точки разрыва и родительские хромосомы обмениваются сегментом, который находится между двумя этими точками	Родитель 1: 100 101101 001 Родитель 2: 010 001100 111 Потомок 1: 100 001100 001 Потомок 2: 010 101101 111
deap.tools .cxUniform	(<i>ind1, ind2, indpb</i>)	Равномерный	каждый бит первого потомка случайным образом наследуется от одного из родителей, второму потомку достается бит другого родителя	Родитель 1: 100101101001 Родитель 2: 010001100111 Вероятность: 90 % Случайные числа (100): 2, 24, 8, 93, 55, 13, 67, 43, 99, 61, 5, 89 Потомок 1: 100 0 0110 0 001 Потомок 2: 010 10110 111
deap.tools .cxOrdered	(<i>ind1, ind2</i>)	Упорядоченный	1) Выбор двух точек разрыва. 2) Обмен центральными частями 2) Обход всех недействующих генов в особи, начиная со второй точки разрыва (гены значения которых не в середине, остаются на месте, повторяющиеся (были в середине), заменяются следующим неповторяющимся значением из этой же особи).	Родитель 1: <u>123 4567 89</u> Родитель 2: <u>375 2814 96</u> Потомок 1: 567 2814 93 Потомок 2: 2814567 93 9 – остался на месте, остальные сдвинулись

Подробнее см. <https://deap.readthedocs.io/en/master/api/tools.html>.

Пример:

```
toolbox.register("mate", tools.cxTwoPoint)
```

Мутация

Существует много методов мутации, в модуле `tools` реализовано 6 (Таблица 4), рассмотрим 3 из них (Таблица 7), для которых нужен следующий набор параметров:

- *individual* – особь,
- *mu* – среднее или последовательность средних для гауссовой аддитивной мутации,
- *sigma* – стандартное отклонение или последовательность стандартных отклонений для гауссовой аддитивной мутации,
- *indpb* – вероятность мутации,
- *eta* – степень скопления мутаций: высокая создаст мутанта, похожего на своего родителя, маленькая эта даст большие отличий,
- *low* – значение или последовательность значений, являющаяся нижней границей пространства поиска,
- *up* – значение или последовательность значений, являющаяся верхней границей пространства поиска.

Таблица 7 – Пример операторов мутации

Функция	Параметры	Описание
<code>deap.tools.mutGaussian</code>	$(individual, mu, sigma, indpb)$	добавляет случайное число, заданное гауссовым распределением с нулевым средним для каждого компонента входного вектора (применяется для вещественного типа)
<code>deap.tools.mutFlipBit</code>	$(individual, indpb)$	Меняет на противоположное значение бит с определённой вероятностью (применяется к логическому типу)
<code>deap.tools.mutPolynomialBounded</code>	$(individual, eta, low, up, indpb)$	Полиномиальная мутация

Подробнее см. <https://deap.readthedocs.io/en/master/api/tools.html>.

Пример:

1) Гауссовская мутация

```
toolbox.register("attr", random.random)
```

```
toolbox.register("mutate", tools.mutGaussian, mu=0.0, sigma=0.2, indpb=0.2)
```

2) Инверсия бит

```
toolbox.register("attr", random.randint, 0, 1)
```

```
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
```

3) Полиномиальная мутация

```
toolbox.register("attr", random.random)
```

```
toolbox.register("mutate", tools.mutPolynomialBounded, eta=10.0, low=0.1,  
up=1, indpb=0.4)
```

Определение популяции

Популяция может быть представлена в виде массива особей, матрицы, роя (алгоритма роя), подпопуляций (для островной модели). Необходимо определить имя в наборе инструментов Toolbox, тип для популяции (список или массив с определённой размерностью), объект, который описывает особи (индивиды), и функцию генерации популяции (это может быть например `tools.initRepeat` как и для генерации особи).

Пример:

```
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

Определение инструментов контроля функционирования генетического алгоритма

Для фиксации эволюционных изменений в популяциях можно использовать готовые классы из модуля `tools` (Таблица 8).

Таблица 8 – Пример классов для хранения данных о функционировании ГА

Класс		Описание
<code>deap.tools.Statistics</code>	<code>([key])</code> – необязательный параметр, идентификатор для доступа к сохраняемым значениям, значение, возвращаемое ключом, может быть многомерным объектом	собирает статистику по списку произвольных объектов, объект хранения регистрируется с помощью метода <i>register</i>
<code>deap.tools.Logbook</code>	нет	Объект - эволюционные записи в виде хронологического списка

		словарей. Данные могут быть получены с помощью метода <code>select</code> .
<code>deap.tools.HallOfFame</code>	<p>(<i>maxsize</i>, <i>similar=<built-in function eq></i>)</p> <p><i>Maxsize</i> – максимальное количество особей в зале славы</p> <p><i>similar</i> - Оператор эквивалентности между двумя особями (необязательный параметр)</p>	Зал славы содержит лучшую особи популяции в процессе эволюции, особи лексикографически отсортированы, первый элемент Зала славы особь с максимальной приспособленностью.

Подробнее см. <https://deap.readthedocs.io/en/master/api/tools.html>.

Пример:

1) Статистика и зал славы

```
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", numpy.mean)
stats.register("std", numpy.std)
stats.register("min", numpy.min)
stats.register("max", numpy.max)
```

2) Книга логов

```
log = Logbook()
log.record(gen=0, mean=5.4, max=10.0)
log.record(gen=1, mean=9.4, max=15.0)
log.select("mean")
log.select("gen", "max")
```

Запуск генетического алгоритма

Первый шаг генетического алгоритма - формирование начальной популяции. Он выполняется зарегистрированным в наборе инструментов методом (вызов) и его значение присваивается новой переменной.

Пример:

```
pop = toolbox.population(n=300)
```

Далее необходимо определить стратегию или модель функционирования генетического алгоритма. В модуле `algorithms` есть готовые реализации стратегий, которые можно использовать (Таблица 9), кроме этого можно запрограммировать работу генетического алгоритма по требуемой модели, используя операторы `rpython` (см. 2 пример). Для использования готовых стратегий необходимо учитывать следующие параметры:

- *population* - популяция,
- *toolbox* – набор инструментов,
- *cxpb* – вероятность кроссовера,
- *mutpb* – вероятность мутации,
- *ngen* – количество поколений,
- *stats* – статистика по ГА (необязательный параметр),
- *halloffame* – «зал славы», лучшие особи в поколениях(необязательный параметр),
- *verbose* – включать ли лог в статистику(необязательный параметр),
- *mu* – количество особей, выбираемых для следующего поколения,
- *lambda_* - количество потомков, порождаемых в каждом поколении.

Таблица 9 – Варианты моделей (стратегий) ГА

Функция	Параметры	Описание
<code>deap.algorithms.eaSimple</code>	<i>(population, toolbox, cxpb, mutpb, ngen[, stats, halloffame, verbose])</i>	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
<code>deap.algorithms.eaMuPlusLambda</code>	<i>(population, toolbox, mu, lambda_, cxpb, mutpb, ngen[, stats, halloffame, verbose])</i>	Только первый ребёнок добавляется в популяцию, второй отбрасывается. Потомок не подвергается мутации (мутируют родители). В $(\mu+\lambda)$ -стратегиях селекция производится из $(\mu+\lambda)$ особей - объединённой популяции родителей и потомков. Необходимо регистрировать функции: <code>toolbox.mate()</code> , <code>toolbox.mutate()</code> , <code>toolbox.select()</code> и <code>toolbox.evaluate()</code> .
<code>deap.algorithms.eaMuCommaLambda</code>	<i>(population, toolbox, mu, lambda_, cxpb, mutpb, ngen[, stats, halloffame, verbose])</i>	Только первый ребёнок добавляется в популяцию, второй отбрасывается. Потомок не подвергается мутации (мутируют родители). В (μ, λ) -стратегиях в каждой итерации происходит генерация λ потомков, из которых

		выбирается μ особей. Необходимо регистрировать функции: toolbox.mate(), toolbox.mutate(), toolbox.select() и toolbox.evaluate().
deap.algorithms.eaGenerateUpdate	(toolbox, ngen[, stats, halloffame, verbose])	Алгоритм генерирует особи с помощью функции toolbox.generate() и изменяет / обновляет их с помощью toolbox.update(). Необходимо регистрировать функции: toolbox.generate(), toolbox.evaluate()

Пример:

1) Простая эволюционная стратегия, используются оба потомка.

```
pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.2, ngen=40,
stats=stats, halloffame=hof, verbose=True)
```

2) Стратегия ($\mu+\lambda$).

```
MU, LAMBDA = 100, 100
```

```
pop, log = algorithms.eaMuPlusLambda(pop, toolbox, mu=MU, lambda_=LAMBDA,
cxpb=0.7, mutpb=0.3, ngen=NGEN, stats=stats, verbose=True, halloffame=hof)
```

Просмотр результатов функционирования генетического алгоритма

Результаты логов, статистики, зала славы можно вывести на печать или на график (см. примеры ниже).

Пример реализации генетического алгоритма

1) Реализация ГА на базе модели eaSimple.

```
import random
import numpy
from deap import algorithms, base, creator, tools
def evalOneMax(individual):
    return sum(individual),

creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_bool, 100)
toolbox.register("evaluate", evalOneMax)
toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", numpy.mean)
stats.register("std", numpy.std)
stats.register("min", numpy.min)
stats.register("max", numpy.max)
```

```

pop = toolbox.population(n=300)
pop, log = algorithms.eaSimple(pop, toolbox, cxpb=0.5, mutpb=0.2, ngen=40,
stats=stats, halloffame=hof, verbose=True)
print(pop, log, hof)

```

2) Реализация ГА без модели (с помощью цикла), использование статистики и зала славы, вывод на график показателей функционирования ГА.

```

# pip install deap
# https://deap.readthedocs.io/en/master/
# Библиотека построения графиков
import matplotlib.pyplot as plt
# библиотека работы со случайными величинами
import random
# библиотека для работы с массивами
import numpy
# библиотека генетического алгоритма
from deap import base, creator, tools
# создаем классы FitnessMax на основе класса base.Fitness и Individual на
основе списка
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)
#Инициализировать панель инструментов
toolbox = base.Toolbox()
toolbox.register("attr_bool", random.randint, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_bool, 10)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
# функция оценки
def evalOneMax(individual):
    return sum(individual),
#Зарегистрировать оператора оценки (фитнес-функцию)
toolbox.register("evaluate", evalOneMax)
# кроссовер
toolbox.register("mate", tools.cxTwoPoint)
# мутация
toolbox.register("mutate", tools.mutFlipBit, indpb=0.05)
# отбор
toolbox.register("select", tools.selTournament, tournsize=3)
n=20
l=numpy.zeros(n, dtype=float)
ll=numpy.zeros(n, dtype=float)
# создаем объект история
history=tools.History()
# Decorate the variation operators
toolbox.decorate("mate", history.decorator)
toolbox.decorate("mutate", history.decorator)
def main():
    #генерируем популяцию, в скобках количество особей в популяции
    pop = toolbox.population(n=30)
    history.update(pop)
    # рассчитываем массив приспособленности
    fitnesses = list(map(toolbox.evaluate, pop))
    # присваиваем значение приспособленности соответствующим особям
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
    fits = [ind.fitness.values[0] for ind in pop]
    #параметры скрещивания и мутации
    CXPB, MUTPB = 0.5, 0.2
    g = 0
    # Функционирование ГА
    while g < n-1:
        g = g + 1
        print("-- Поколение %i --" % g)

```



```

# выбираем особи следующего поколения
offspring = toolbox.select(pop, len(pop))
# клонируем, чтобы образовывать пары
offspring = list(map(toolbox.clone, offspring))
# применяем кроссовер
for child1, child2 in zip(offspring[::2], offspring[1::2]):
    if random.random() < CXPB:
        # скрещиваем 2 особи
        toolbox.mate(child1, child2)
        # удаляем скрещенные особи
        del child1.fitness.values
        del child2.fitness.values
# применяем мутацию
for mutant in offspring:
    if random.random() < MUTPB:
        toolbox.mutate(mutant)
        # удаляем немутированный вариант особи
        del mutant.fitness.values
# Проверка валидности значения приспособленности
invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
fits = map(toolbox.evaluate, invalid_ind)
for ind, fit in zip(invalid_ind, fits):
    ind.fitness.values = fit
# заменяем популяцию на новое поколение
pop[:] = offspring
history.update(pop)
# расчет приспособленности новой популяции
fits = [ind.fitness.values[0] for ind in pop]
length = len(pop)
mean = sum(fits) / length
sum2 = sum(x*x for x in fits)
std = abs(sum2 / length - mean**2)**0.5
l[g] = mean
ll[g] = g
print(" Минимальная приспособленность %s" % min(fits))
print(" Максимальная приспособленность %s" % max(fits))
print(" Среднее значение %s" % mean)
print(" Std %s" % std)
# просмотр лучшей особи
best_ind = tools.selBest(pop, 1)[0]
print("Лучшая особь %s, %s" % (best_ind, best_ind.fitness.values))

main()
plt.plot(ll, l, linewidth=2.0)

```

Варианты для выполнения лабораторной работы

	Целевая функция	Вид особи	Приспособленность	Отбор	Кроссовер	Мутация	Стратегия
1	Максимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 5 ген особи	Турнирный отбор	Одноточечный	Полиномиальная мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
2	Минимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Отбор рулеткой	Упорядоченный	Меняет на противоположное значение ген	($\mu+\lambda$)
3	Максимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции	Выбор случайных k особей	Двухточечный	Гауссовская мутация	(μ, λ)
4	Минимум, однокритериальная	Тип: первый параметр булевый, длина особи: 7	Количество элементов =1 (истина)	Выбор лучших k особей	Равномерный	Меняет на противоположное значение ген	($\mu+\lambda$)
5	Минимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторяться, длина особи: 10	Среднее значение генов особей популяции	Выбор лучших k особей	Равномерный	Гауссовская мутация	($\mu+\lambda$)
6	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Выбор случайных k	Упорядоченный	Меняет на противоположное значение ген	Оба потомка заменяют родителей. Мутировать могут и родители и

				особей			потомки.
7	Минимум, однокритериальная	Тип: массив вещественных чисел (numru), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 2 ген особи	Отбор рулеткой	Двухточечный	Полиномиальная мутация	(μ, λ)
8	Максимум, однокритериальная	Тип: целочисленный в интервале от 0 до 5 могут повторяться, длина особи: 17	сумма значений генов	Турнирный отбор	Равномерный	Меняет на противоположное значение ген	($\mu+\lambda$)
9	Максимум, однокритериальная	Тип: массив вещественных чисел (agray), значения: от 0 до 1 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 5 ген особи	Турнирный отбор	Двухточечный	Полиномиальная мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
10	Минимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Выбор лучших k особей	Упорядоченный	Меняет на противоположное значение ген	($\mu+\lambda$)
11	Максимум, однокритериальная	Тип: массив вещественных чисел (numru), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции	Выбор случайных k особей	Равномерный	Гауссовская мутация	(μ, λ)
12	Минимум, однокритериальная	Тип: булевый длина особи: 10	Количество элементов =1 (истина) минус 5 ген	Выбор случайных k особей	Одноточечный	Полиномиальная мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
13	Минимум, однокритериальная	Тип: массив вещественных чисел (agray), значения: от 0 до 1 могут	Среднее значение генов особей популяции	Турнирный отбор	Одноточечный	Гауссовская мутация	($\mu+\lambda$)

		повторятся, длина особи: 10					
14	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Турнирный отбор	Упорядоченный	Меняет на противоположное значение ген	$(\mu+\lambda)$
15	Минимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 2 ген особи	Турнирный отбор	Равномерный	Гауссовская мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
16	Максимум, однокритериальная	Тип: целочисленный в интервале от 0 до 5 длина особи: 7	1) Количество элементов =1 (истина)	Турнирный отбор	Одноточечный	Меняет на противоположное значение ген	$(\mu+\lambda)$
17	Максимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 5 ген особи	Выбор случайных k особей	Двухточечный	Гауссовская мутация	(μ, λ)
18	Минимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Выбор лучших k особей	Упорядоченный	Меняет на противоположное значение ген	$(\mu+\lambda)$
19	Максимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции	Турнирный отбор	Одноточечный	Гауссовская мутация	(μ, λ)
20	Минимум, однокритериальная	Тип: вещественный в интервале от 0 до, значения не повторяются, длина особи: 14	среднее значение ген	Отбор рулеткой	Двухточечный	Полиномиальная мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
21	Минимум, однокритериальная	Тип: массив вещественных чисел	Среднее значение генов	Отбор	Одноточечный	Гауссовская мутация	$(\mu+\lambda)$

	однокритериальная	(аgгау), значения: от 0 до 1 могут повторяться, длина особи: 10	особей популяции	рулеткой	ый	мутация	
22	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторяться, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Выбор лучших к особей	Равномерны й	Меняет на противоположное значение ген	(μ , λ)
23	Минимум, однокритериальная	Тип: массив вещественных чисел (numru), значения: от 0 до 13 могут повторяться, длина особи: 10	Среднее значение генов особей популяции делённая на 2 ген особи	Выбор случайных к особей	Одноточечный	Гауссовская мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.

Тема 5. Нечеткие вычисления.

Задание лабораторной работы

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 принадлежности лингвистических переменных.
4. Поверхности нечёткого вывода.
5. Результаты нечёткого вывода (3 прогона).
6. Код.

Методические указания по выполнению лабораторной работы

Схема построения нечёткого контроллера

Библиотека Skfuzzy содержит набор инструментов Fuzzy Logic для языка Python. Большая часть функциональности находится в подпакетах (см. Таблица 10), но, как numpy, часть основных вынесено функций в базовое пространство имён (см. подробно <https://pythonhosted.org/scikit-fuzzy/api/api.html>).

Таблица 10 – 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 4).

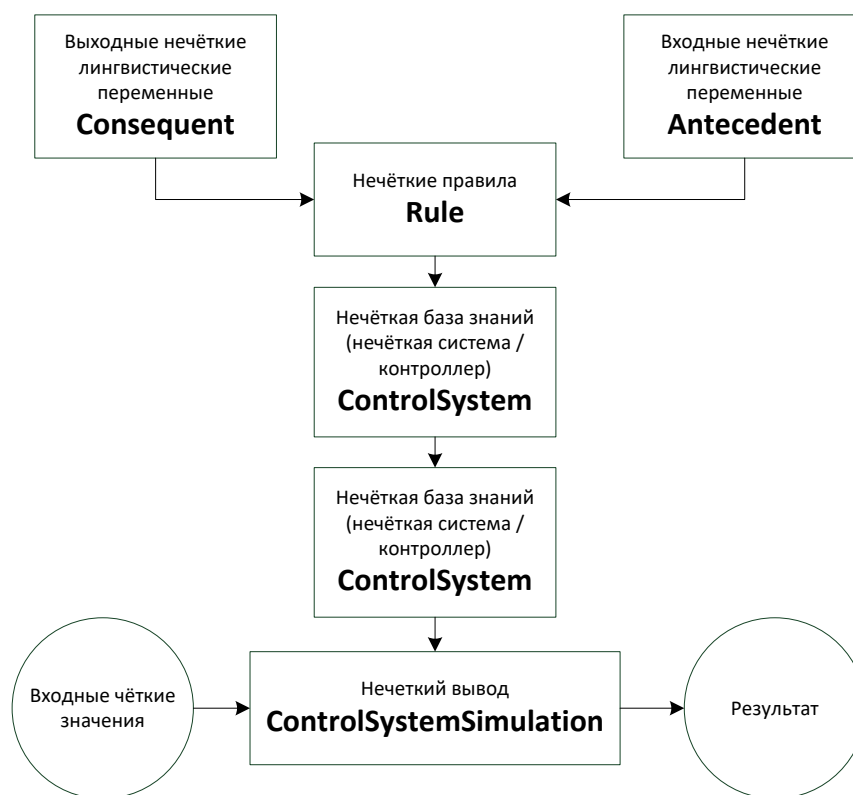


Рисунок 4 – Схема применения классов пакета Control

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 определить 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 11).

Таблица 11 – Примеры методы генерации функций принадлежности (см. подробнее <https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.membership.html>)

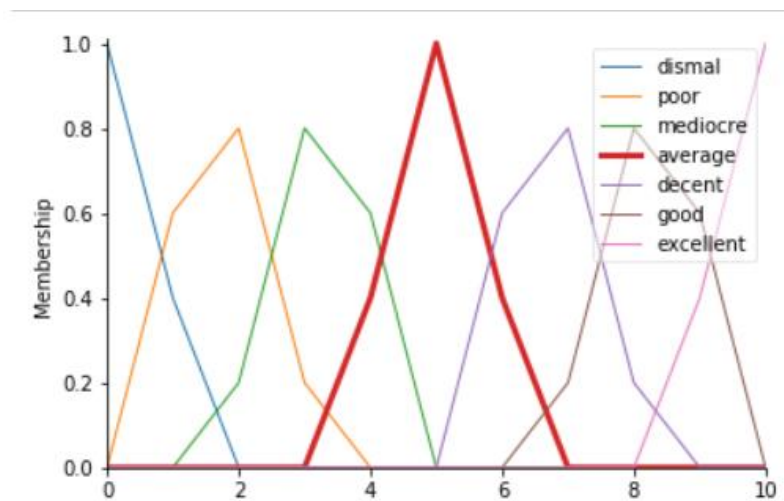
Метод	Описание	Формула
<code>skfuzzy.membership.dsigmf(x, b1, c1, b2, c2)</code>	Разница двух нечётких сигмоидальных функций принадлежности.	$y = f1 - f2$ $f1(x) = 1 / (1. + \exp[- c1 * (x - b1)])$ $f2(x) = 1 / (1. + \exp[- c2 * (x - b2)])$
<code>skfuzzy.membership.gbellmf(x, a, b, c)</code>	Генератор нечеткого членства обобщенной функции Белла.	$y(x) = 1 / (1 + \text{abs}([x - c] / a) ** [2 * b])$
<code>skfuzzy.membership.piecemf(x, abc)</code>	Кусочно-линейная функция принадлежности	$y = 0, \min(x) \leq x \leq a$ $y = (x - a) / (b - a), a \leq x \leq b$ $y = (c - x) / (c - b), b \leq x \leq c$
<code>skfuzzy.membership.sigmf(x, b, c)</code>	The basic sigmoid membership function generator.	$y = 1 / (1. + \exp[- c * (x - b)])$
<code>skfuzzy.membership.trapmf(x,</code>	Трапециевидный генератор	См. Рисунок 5, б.

abcd)	функций принадлежности.	
skfuzzy.membership.trimf(x, abc)	Треугольная функция принадлежности	См. Рисунок 5, с.

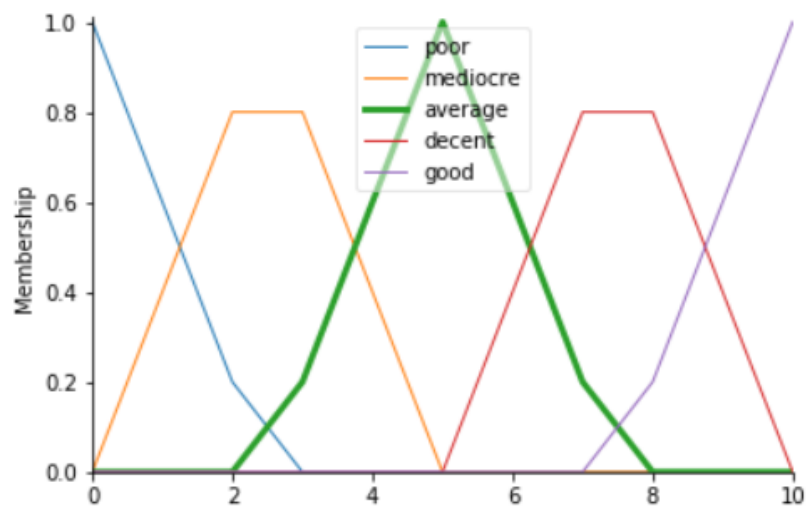
Используются методы генерации, например так:

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

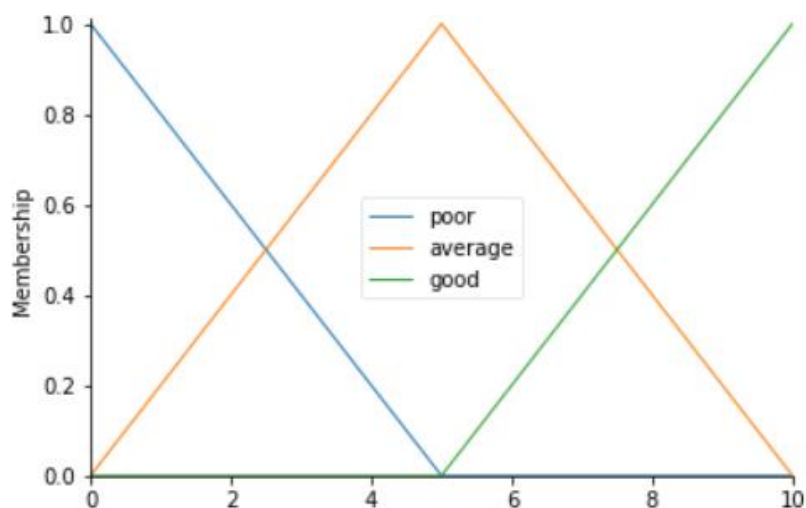
Автоматическое разбиение интервала. При построении функций принадлежности можно использовать автоматическую разбивку интервала на симметричные значения по 3, 5 или 7 термов (Рисунок 5). Имена при этом могут быть заданы автоматически: `dismal`, `poor`, `mediocre`, `average`, `decent`, `good`, `excellent` 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 термина в лингвистической переменной

Рисунок 5 – Графики функций принадлежности при автоматической разбивке с помощью функции `automf`

Визуализация лингвистических переменных

Чтобы вывести график функций лингвистической переменной (Рисунок 5,) необходимо вызвать метод `view()`, для объектов классов `Antecedent` или `Consequent` (`Имя_объекта.view()`). Причём, если необходимо выделить конкретный терм, то необходимо указать имя термина (`Имя_объекта['имя термина'].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 12)

Таблица 12 – Логические операторы в правилах

Логический оператор	Обозначение	Пример
Или		<code>.Rule(in_p1['poor'] in_p2['poor'], out_p['low'])</code>
И	&	<code>.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')</code>
Не	~	<code>.Rule(~ in_p['poor'], out_p['high'])</code>

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 вывода 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 для выполнения лабораторной работы

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 осадков, 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).

Тема 6. Нейронные сети.

Задание лабораторной работы

Цель работы: получение практических навыков программирования нейронных сетей на языке Python с использованием библиотеки PyTorch.

Задание: используя программу Jupiter Notebook, язык программирования Python, библиотеку PyTorch построить нейронную сеть по варианту и использовать для получения результата.

Работа заключается в:

- Загрузке / генерации данных для обучения НС;
- Построения НС;
- Обучения НС;
- Проверки НС на тестовых данных;
- Визуализация результата.

Отчёт по лабораторной работе должен содержать:

1. Фамилию и номер группы учащегося, задание, вариант
2. Схему НС (ее слоёв)
3. Описание входных данные

4. Описание алгоритма обучения с учетом варианта (функции потерь, оптимизатора и т.д.)
5. Графики динамики обучения НС.
6. Результат тестирования НС.
7. Код.

Методические указания по выполнению лабораторной работы

Пакет torch.nn

Пакет torch.nn используется для создания нейронных сетей. Он содержит контейнеры для НС, в котором определяются слои, функции потерь, активации, различные методы оптимизации для реализации обучения и т.д.

Пакет nn определяет набор модулей, которые примерно эквивалентны слоям нейронной сети. Модуль принимает входные Tensors и вычисляет выходные Tensors, но может также содержать внутреннее состояние, такое как Tensors, содержащее обучаемые параметры.

Алгоритм работы с НС

- 1 Подготовка данных для обучения /анализа (обучающая выборка), их преобразование.
- 2 Выбирается тип НС в зависимости от поставленной задачи (прямого распространения, рекуррентная, сверточная и т.д.), выбирается архитектура сети (количество слоёв, нейронов в слоях, типы слоёв, функции активации), строится модель.
- 3 Определение функции потерь.
- 4 Определение оптимизатора.
- 5 Цикл обучения НС (на примере сети прямого распространения)
 - ввод данных и вычисление результата (прямой проход)
 - вычисление потери (насколько далёк результат от правильности).
 - градиентный спуск и коррекция весов (обратный проход).
- 6 Запуск / тестирование НС (на тестовой выборке).

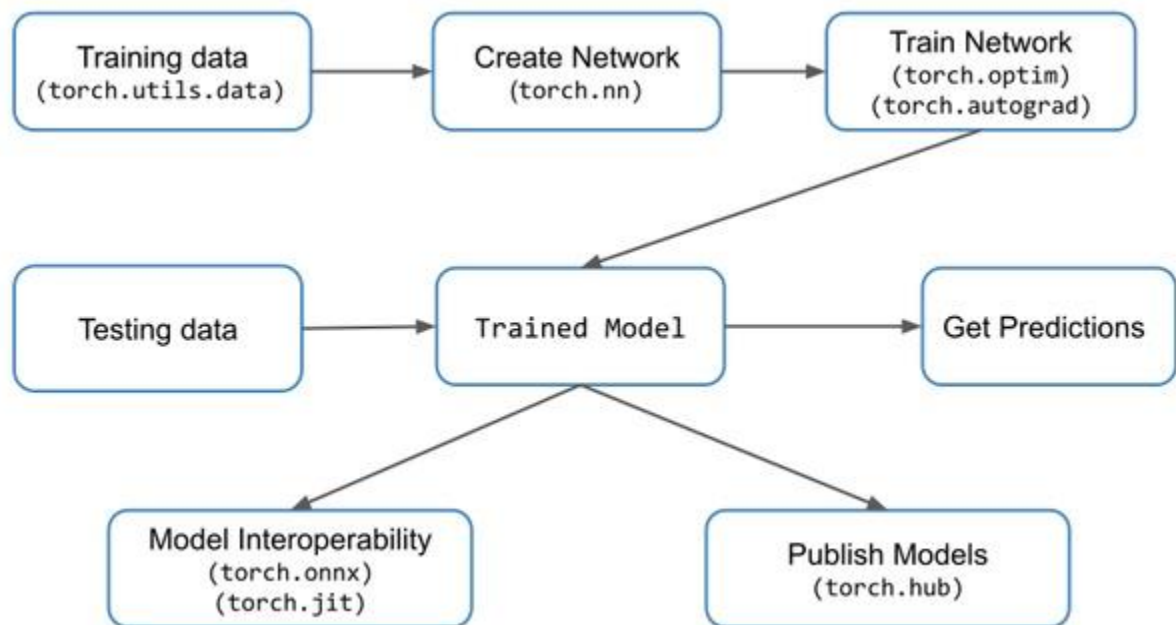


Рисунок 6 – Схема применения модулей PyTorch для обучения НС²

Загрузка подготовленного набора данных

PyTorch включает в себя пакет torchvision, который используется для загрузки и подготовки набора данных (<https://pytorch.org/docs/master/torchvision/index.html>). Он включает в себя две основные функции, а именно Dataset и DataLoader, которые помогают в преобразовании и загрузке набора данных.

Dataset построен поверх тензорного типа данных и используется в основном для пользовательских наборов данных. Набор данных используется для чтения и преобразования точки данных из данного набора данных.

Dataset - абстрактный класс, представляющий набор данных. Пользовательский набор данных должен наследовать Dataset и переопределять следующие методы:

- `__len__`, чтобы `len(dataset)` возвращал размер набора данных.
- `__getitem__` для поддержки индексации, так что `dataset[i]` может использоваться для получения i-го экземпляра

Основной синтаксис для реализации упомянут ниже:

```
trainset = torchvision.datasets.CIFAR10(root = './data', train = True,
download = True, transform = transform)
```

Пример:

1) Загрузка набора данных MNIST

```
import torchvision
```

² https://ai-news.ru/2019/07/pytorch_dlya_nachinaushih_osnovy.html

```
train_dataset = torchvision.datasets.MNIST(root='g:\\DataForNN2', train=True,
transform=False, download=True)
```

3) Загрузка набора CIFAR10

```
trainset = torchvision.datasets.CIFAR10(root = DATA_PATH, train = True,
download = True, transform = False)
```

4) Загрузка набора STL10

```
torchvision.datasets.STL10(DATA_PATH, split='train', folds=None,
transform=None, target_transform=None, download=True)
```

Далее необходимо создать объекты `train_dataset` и `test_dataset`, которые будут последовательно проходить через загрузчик данных. Чтобы создать такие датасеты из данных MNIST, требуется задать несколько аргументов. Первый — путь до папки, где хранится файл с данными для тренировки и тестирования. Логический аргумент `train` показывает, какой файл из `train.pt` или `test.pt` стоит брать в качестве тренировочного сета. Следующий аргумент — `transform`, в котором мы указываем ранее созданный объект `trans`, который осуществляет преобразования. Наконец, аргумент загрузки просит функцию датасета MNIST загрузить при необходимости данные из онлайн источника.³

Таблица 13 – Примеры наборов данных для обучения (полный список см. <https://pytorch.org/docs/stable/torchvision/datasets.html>)

MNIST	Рукописные цифры 1–9. Подмножество набора данных NIST рукописных символов. Содержит обучающий набор из 60000 тестовых изображений и тестовый набор из 10000.
Fashion-MNIST	Набор данных для MNIST. Содержит изображения предметов моды; например, футболка, брюки, пуловер.
EMNIST	На основе рукописных символов NIST, включая буквы и цифры и разделение для задач классификации классов 47, 26 и 10.
COCO	Более 100 000 изображений, классифицированных в повседневные предметы; например, человек, рюкзак и велосипед. Каждое изображение может иметь более одного класса.
LSUN	Используется для крупномасштабной классификации сцен изображений; например, спальня, мост, церковь.
Imagenet-12	Крупномасштабный набор данных визуального распознавания,

³ <https://neurohive.io/ru/tutorial/cnn-na-pytorch/>

	содержащий 1,2 миллиона изображений и 1000 категорий. Реализовано с классом ImageFolder, где каждый класс находится в папке.
CIFAR	60 000 цветных изображений с низким разрешением (32 32) в 10 взаимоисключающих классах; например, самолет, грузовик и автомобиль.
STL10	Аналогично CIFAR, но с более высоким разрешением и большим количеством немаркированных изображений.
SVHN	600 000 изображений номеров улиц, полученных из Google Street View. Используется для распознавания цифр в реальных условиях.
PhotoTour	Изучение локальных дескрипторов изображений. Состоит из полутоновых изображений, состоящих из 126 фрагментов, сопровождаемых текстовым файлом дескриптора. Используется для распознавания образов.

Преобразование данных

Функция `transform.Compose()` находится в пакете `torchvision` и позволяет выполнять трансформацию набора данных, причём трансформаций может быть несколько и они представляются списком.

Пример:

1) Загрузка MNIST с преобразованием

```
import torchvision
import torchvision.transforms as transforms
# путь куда грузим
DATA_PATH = 'g:\\DataForNN'
# выполняемое преобразование над набором данных
trans = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.1307,), (0.3081,))])
# грузим набор данных тренировочный
train_dataset = torchvision.datasets.MNIST(root=DATA_PATH, train=True,
transform=trans, download=True)
# грузим набор данных тестовый
test_dataset = torchvision.datasets.MNIST(root=DATA_PATH, train=False,
transform=trans)
```

В примере устанавливается преобразование, которое конвертирует входной датасет в PyTorch тензор. PyTorch тензор — особый тип данных, используемый в библиотеке для всех различных операций с данными и весами внутри нейросети. Следующий аргумент в списке `Compose()` — нормализация. Нейронная сеть обучается лучше, когда входные

данные нормализованы так, что их значения находятся в диапазоне от -1 до 1 или от 0 до 1. Чтобы это сделать с помощью нормализации PyTorch, необходимо указать среднее и стандартное отклонение MNIST датасета, которые в этом случае равны 0.1307 и 0.3081 соответственно. У MNIST есть только один канал, но уже для датасета CIFAR с 3 каналами (по одному на каждый цвет из RGB спектра) надо указывать среднее и стандартное отклонение для каждого.

Загрузка данных для тренировки нейронной сети

DataLoader используется, когда есть большой набор данных, и необходимо загрузить данные из Dataset в фоновом режиме, чтобы он был готов и ждал цикла обучения.

DataLoader используется для перемешивания и пакетной обработки данных. Он может использоваться для загрузки данных параллельно с многопроцессорными рабочими. Объект загрузчик данных в PyTorch обеспечивает несколько полезных функций при использовании тренировочных данных:

- Возможность легко перемешивать данные.
- Возможность группировать данные в партии.
- Более эффективное использование данных с помощью параллельной загрузки, используя многопроцессорную обработку.

Синтаксис:

```
trainloader = torch.utils.data.DataLoader(trainset, batch_size = 4,
    shuffle = True)
```

Первый — данные, которые вы хотите загрузить; второй — желаемый размер партии; третий — перемешивать ли случайным образом датасет.

Построение нейронной сети

На базе *nn.Module*. Необходимо наследовать класс *nn.Module* и реализовать методы инициализации *init* и прямого прохода/вычисления *forward*. Синтаксис следующий:

```
# подключаем модуль torch.nn
import torch.nn as nn
# импортируем функции активации
import torch.nn.functional as F
# Model это имя
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
    ...
```

```
def forward(self, x):
```

```
...
```

Внутри функции `__init__` объявляют слои будущей нейронной сети, они могут быть разными по тиру, в зависимости от того, какую нейронную сеть строим, количество и размерность слоёв определяется здесь же. Размерность следующих друг за другом слоев должна быть согласована, сколько выходов в предшествующем, столько входов в последующем.

Пример:

1) Два линейных слоя (нейронная сеть прямого распространения, в которой слои полносвязные), которые имеют вид $\mathbf{W}\mathbf{x}+\mathbf{b}$, где \mathbf{W} – матрица весов размером $(input, output)$ и \mathbf{b} – вектор смещения размером $output$. Первый слой размерностью $(784, 100)$, второй $(100, 10)$, т.е. выходной вектор НС размерностью 10:

```
class Model(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(784, 100)
        self.fc2 = nn.Linear(100, 10)
```

2) Два слоя двумерной свёртки, первый слой имеет 1 входной канал, 20 выходных и размер ядра 5, второй, 20 входных :

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.conv1 = nn.Conv2d(1, 20, 5)
        self.conv2 = nn.Conv2d(20, 20, 5)
```

3) Свёрточная сеть с 2 сверточными слоями и 3 линейными (полносвязными):

```
class Model(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, 3)
        self.conv2 = nn.Conv2d(6, 16, 3)
        self.fc1 = nn.Linear(16 * 6 * 6, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

4) Свёрточная сеть Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear.

```
class MNISTConvNet(nn.Module):
    def __init__(self):
```

```

super(MNISTConvNet, self).__init__()
self.conv1 = nn.Conv2d(1, 10, 5)
self.pool1 = nn.MaxPool2d(2, 2)
self.conv2 = nn.Conv2d(10, 20, 5)
self.pool2 = nn.MaxPool2d(2, 2)
self.fc1 = nn.Linear(320, 50)
self.fc2 = nn.Linear(50, 10)

```

5) Рекуррентная нейронная сеть.

```

class RNNModel(nn.Module):
    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
        super(RNNModel, self).__init__()
        self.hidden_dim = hidden_dim
        self.layer_dim = layer_dim
        self.rnn = nn.RNN(input_dim, hidden_dim, layer_dim, batch_first=True,
                           nonlinearity='relu')
        self.fc = nn.Linear(hidden_dim, output_dim)

```

Метод *forward* используется непосредственно для преобразования входных данных с помощью заданной нейронной сети в ее выходы.

Вычисляемая функция может быть любой сложности, но должна учитывать заданные слои в функции *init*.

Пример:

1) Определение для линейных слоёв из примеры выше. Функция *view()* переиндексирует тензор с данными заданным образом, "-1" в качестве первого аргумента функции означает, что количество элементов в первой размерности будет вычислено автоматически. Если исходный тензор *x* имеет размерность (N, 28, 28), то после *x = x.view(-1, 28*28)* его размерность станет равна (N, 784).

```

def forward(self, x):
    x = x.view(-1, 28*28)
    x = F.relu(self.fc1(x))
    x = self.fc2(x)
    x = F.softmax(x, dim=1)
    return x

```

2) Для 2 примеры выше, обращаемся по определённым ранее именам слоёв, используется функция *relu*:

```

def forward(self, x):
    x = F.relu(self.conv1(x))

```

```
return F.relu(self.conv2(x))
```

3) Пример для сверточной сети

```
def forward(self, x):  
    x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))  
    x = F.max_pool2d(F.relu(self.conv2(x)), 2)  
    x = x.view(-1, self.num_flat_features(x))  
    x = F.relu(self.fc1(x))  
    x = F.relu(self.fc2(x))  
    x = self.fc3(x)  
    return x
```

4) Пример для свёрточной сети Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear

```
def forward(self, input):  
    x = self.pool1(F.relu(self.conv1(input)))  
    x = self.pool2(F.relu(self.conv2(x)))  
    x = x.view(x.size(0), -1)  
    x = F.relu(self.fc1(x))  
    x = F.relu(self.fc2(x))  
    return x
```

4) Пример для рекуррентной сети.

```
def forward(self, x):  
    h0 = Variable(torch.zeros(self.layer_dim, x.size(0), self.hidden_dim))  
    out, hn = self.rnn(x, h0)  
    out = self.fc(out[:, -1, :])  
    return out
```

Для построения модели надо создать объект описанного класса:

```
Net=Model()
```

На базе *nn.Sequential*. Контейнер для линейных / последовательных слоёв *Linear*, которые имеют вид $\mathbf{W}\mathbf{x}+\mathbf{b}$, где \mathbf{W} — матрица весов размером (*input*, *output*) и \mathbf{b} — вектор смещения размером *output*.

Используя этот контейнер можно в одном операторе определить и вид НС и как будут вычисляться выходные значения.

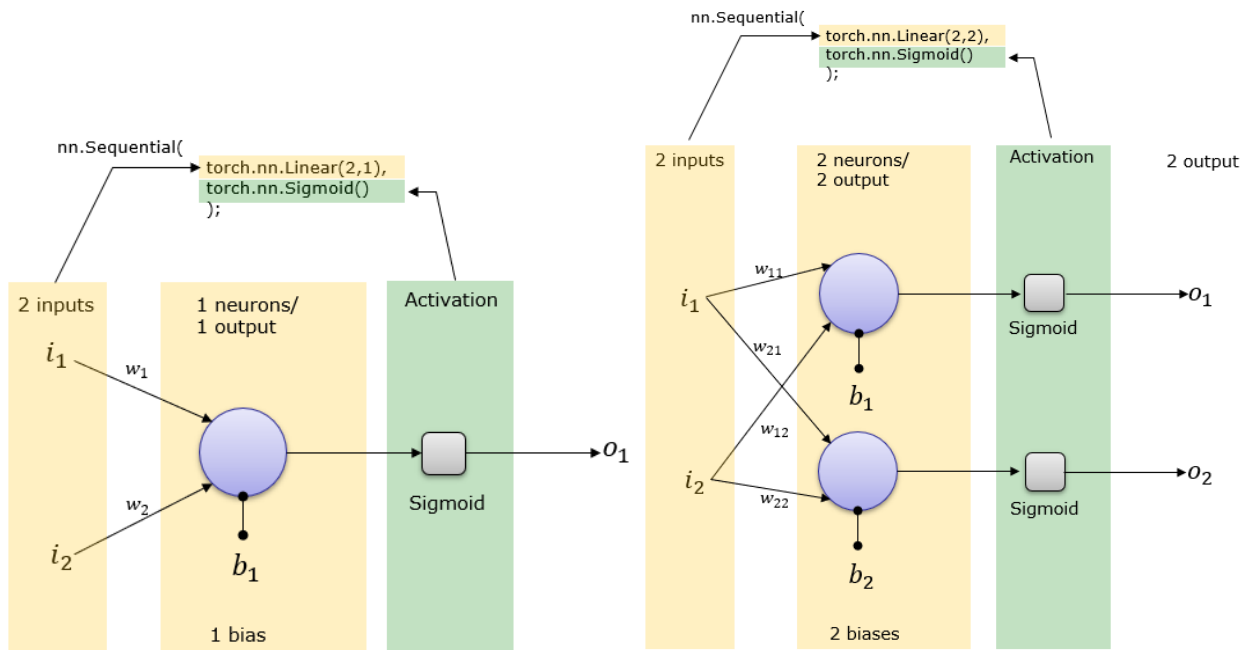
Пример:

НС с 10 входами, с функцией ReLU(), 5 нейронов в скрытом слое, выходной нейрон 1 с функцией сигмоида.

```
model = nn.Sequential(nn.Linear(10, 5),  
    nn.ReLU(),
```

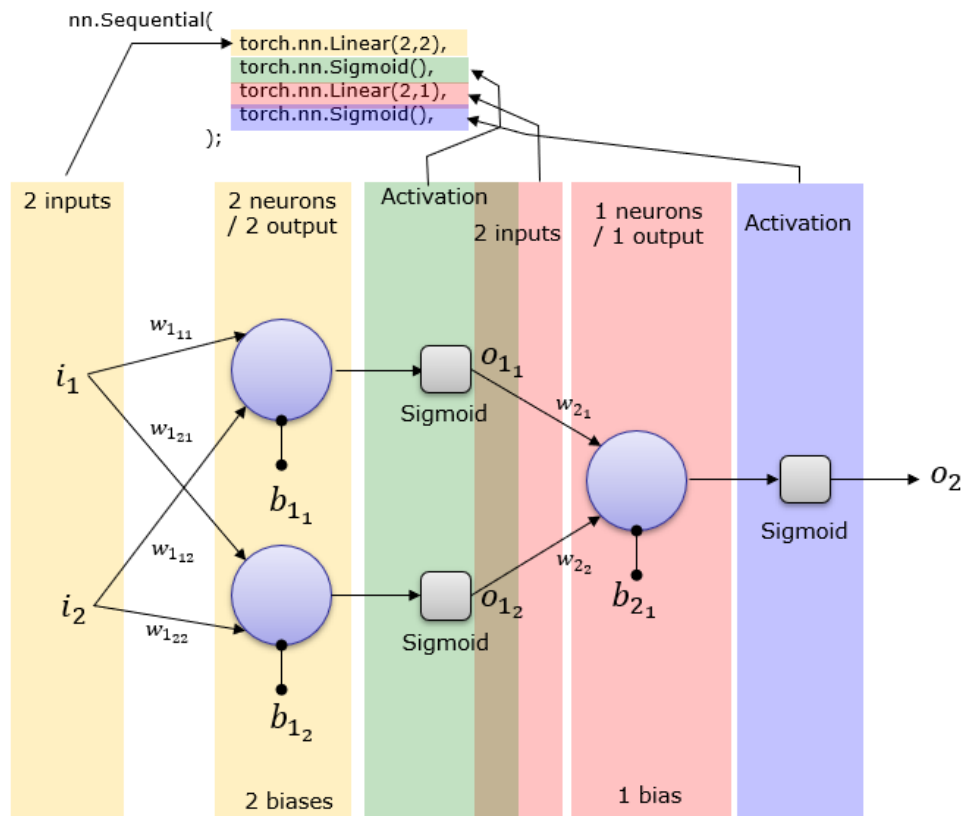


```
nn.Linear(5, 1),
nn.Sigmoid())
```



а) из 2 входов и 1 выхода

б) 2 входа 2 выхода



в) 2 слоя в первом 2 нейрона во втором 1, функция активации - сигмоида

Рисунок 7 - Примеры простейшего линейного слоя⁴

⁴ http://www.sharetechnote.com/html/Python_PyTorch_nn_Sequential_01.html

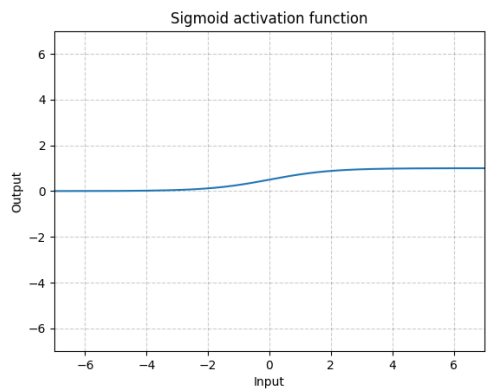
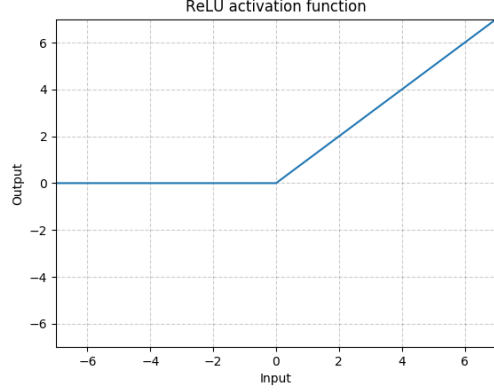
Существуют ещё другие возможности по построению НС. В зависимости от вида НС слои и функции активации могут быть разными., см. подробнее <https://pytorch.org/docs/stable/nn.html>.

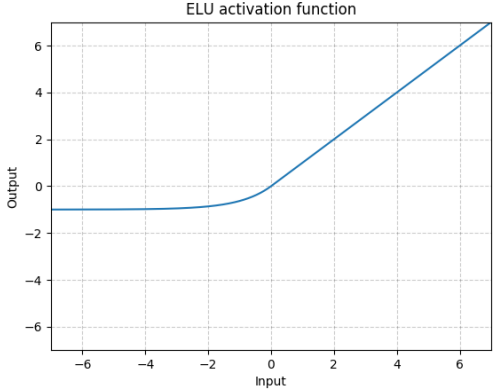
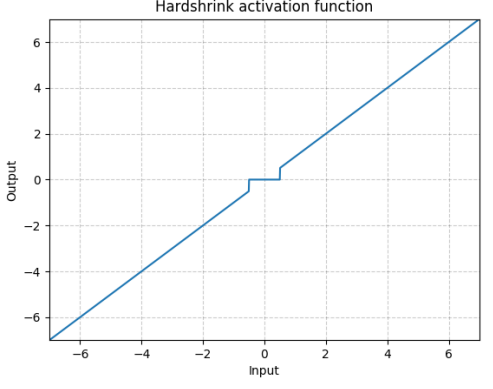
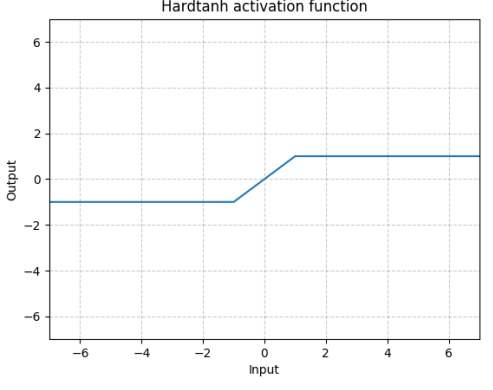
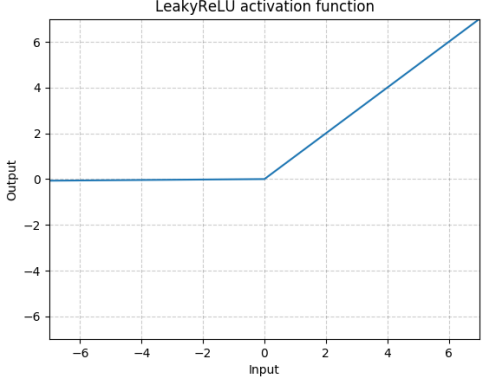
Таблица 14 – Примеры типов слоёв НС

Типы слоев	Назначение	Описание
Активация (activation)	Общие для НС	Содержит функцию активации, которую применяют ко входам слоя.
Нормализация (Normalization)		Слой обеспечивает применение градиентного спуска не к одной точке выборки, а к небольшой коллекции данных.
Прореживание (dropout), регуляризация		Слой обеспечивает добавление информации к условию с целью предотвращения переобучения.
Рекуррентные (Recurrent)	Для рекуррентных НС	Основной блок рекуррентных НС
Свёртка (Convolution)	Для свёрточных НС	основной блок свёрточной нейронной сети, включает в себя для каждого канала свой фильтр, ядро свёртки которого обрабатывает предыдущий слой по фрагментам (суммируя результаты поэлементного произведения для каждого фрагмента).
Пулинг / группировка / субдискретизация (Pooling)		слои пулинга, как правило, чередуются со слоями свёртки и обобщает (упрощает) информацию, полученную от слоя свёртки, пулинг «сжимает» карты признаков, полученные на предыдущем сверточном слое.
Дополнение отступа (Padding)		пиксели, которые находятся на границе изображения участвуют в меньшем количестве сверток, чем внутренние. В связи с этим в сверточных слоях используется дополнение изображения (англ. padding). Выходы с предыдущего слоя дополняются пикселями так, чтобы после свертки сохранился размер

		изображения. Такие свертки называют одинаковыми (same convolution), а свертки без дополнения изображения называются правильными (valid convolution).
Линейный / Соединение «все-со- всеми» / полносвязный (Linear)	Для НС с прямым распространением и свёрточных	Все входы связаны со всеми нейронами слоя, используются в НС прямого распространения, как последний слой свёрточной сети.

Таблица 15 – Пример функций активации нейронов (подробнее см. <https://pytorch.org/docs/stable/nn.html#non-linear-activations-weighted-sum-nonlinearity>)

Функция активации	Метод функции / формула	График функции
Sigmoid	<code>torch.nn.Sigmoid()</code> $\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$	
ReLU	<code>torch.nn.ReLU(inplace=False)</code> $\text{ReLU}(x) = \max(0, x)$	

<p>ELU</p>	<p><code>torch.nn.ELU(alpha=1.0, inplace=False)</code></p> $\text{ELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x) - 1))$	 <p>The graph shows the ELU activation function. The x-axis is labeled 'Input' and ranges from -6 to 6. The y-axis is labeled 'Output' and ranges from -6 to 6. The function is zero for x >= 0 and follows the curve y = exp(x) - 1 for x < 0.</p>
<p>Hardshrink k</p>	<p><code>torch.nn.Hardshrink(lambd=0.5)</code></p> $\text{HardShrink}(x) = \begin{cases} x, & \text{if } x > \lambda \\ x, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases}$	 <p>The graph shows the Hardshrink activation function with lambda = 0.5. The x-axis is labeled 'Input' and ranges from -6 to 6. The y-axis is labeled 'Output' and ranges from -6 to 6. The function is zero for -0.5 <= x <= 0.5 and follows the identity function y = x for x > 0.5 and x < -0.5.</p>
<p>Hardtanh</p>	<p><code>torch.nn.Hardtanh(min_val=-1.0, max_val=1.0, inplace=False, min_value=None, max_value=None)</code></p> $\text{HardTanh}(x) = \begin{cases} 1 & \text{if } x > 1 \\ -1 & \text{if } x < -1 \\ x & \text{otherwise} \end{cases}$	 <p>The graph shows the Hardtanh activation function. The x-axis is labeled 'Input' and ranges from -6 to 6. The y-axis is labeled 'Output' and ranges from -6 to 6. The function is constant at 1 for x > 1, constant at -1 for x < -1, and follows the identity function y = x for -1 <= x <= 1.</p>
<p>LeakyReLU</p>	<p><code>torch.nn.LeakyReLU(negative_slope=0.01, inplace=False)</code></p> $\text{LeakyReLU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative_slope} \times x, & \text{otherwise} \end{cases}$	 <p>The graph shows the LeakyReLU activation function with a negative slope of 0.01. The x-axis is labeled 'Input' and ranges from -6 to 6. The y-axis is labeled 'Output' and ranges from -6 to 6. The function is zero for x >= 0 and follows the line y = 0.01 * x for x < 0.</p>

LogSigmoid	<code>torch.nn.LogSigmoid()</code>	
d	$\text{LogSigmoid}(x) = \log \left(\frac{1}{1 + \exp(-x)} \right)$	

Обучение нейронной сети

Если в качестве обучения используется градиентный спуск (алгоритм обратного распространения ошибки), то процесс обучения выполняется итерационно и включает прямой проход и обратный.

Прямой проход - вычисление выходов НС и текущей ошибки (функции потерь).

Для вычисления выходных значений НС необходимо, подать на входы НС данные из обучающей выборки и последовательно проходить все слои НС.

В библиотеке `pytorch` в класса `Model` за выполнение этого действия отвечает метод `forward`, именно в нем задаётся схема вычисления выходов НС.

Пример определения метода `forward` (см. выше определение НС):

```
def forward(self, input):
    x = self.pool1(F.relu(self.conv1(input)))
    x = self.pool2(F.relu(self.conv2(x)))
    x = x.view(x.size(0), -1)
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    return x
```

В `PyTorch` необязательно вызывать метод в явном виде, при создании объекта НС с указанием входного тензора, метод вызывается. Для вычисления ошибки (функции потерь), необходимо знать текущие значения выходов НС и ожидаемые (те, на которых обучаем). Существует несколько вариантов расчёта потерь (Таблица 16).

Таблица 16 – Примеры функций потерь (подробнее см.

<https://pytorch.org/docs/stable/nn.html#loss-functions>)

Название функции	Синтаксис	Формула
В зависимости от <code>reduction='none' 'mean' 'sum'</code>		

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top,$$

$$\ell(x, y) = \begin{cases} \text{mean}(L), \\ \text{sum}(L), \end{cases}$$

Если для поля `size_average` установлено значение `False`, потери суммируются для каждой мини-партии. Игнорируется, когда уменьшить является ложным. По умолчанию: `True`

L1Loss	<code>torch.nn.L1Loss(size_aver- age=None, reduce=None, reduction='mean')</code>	$l_n = x_n - y_n ,$
MSELoss (средняя квадратичная)	<code>torch.nn.MSELoss(size_a- verage=None, reduce=None, reduction='mean')</code>	$l_n = (x_n - y_n)^2$
KLDivLoss (информационного расхождения Кульбака-Лейблера)	<code>torch.nn.KLDivLoss(size- _average=None, reduce=None, reduction='mean')</code>	$l_n = y_n \cdot (\log y_n - x_n)$
BCELoss (бинарной кросс-энтропии)	<code>torch.nn.BCELoss(weigh- t=None, size_average=None, reduce=None, reduction='mean')</code>	$l_n = -w_n [y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)],$
BCEWithLogitsLoss	<code>torch.nn.BCEWithLogit- sLoss(weight=None, size_average=None, reduce=None, reduction='mean', pos_weight=None)</code>	$l_n = -w_n [y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n))],$
HingeEmbeddingLoss	<code>torch.nn.HingeEmbeddi- ngLoss(margin=1.0, size_average=None, reduce=None,</code>	$l_n = \begin{cases} x_n, & \text{if } y_n = 1, \\ \max\{0, \Delta - x_n\}, & \text{if } y_n = -1, \end{cases}$

	<code>reduction='mean')</code>	
--	--------------------------------	--

Пример определения функции потерь:

```
loss_fn = torch.nn.MSELoss(reduction='sum')
```

После того как определено, что НС ещё не обучена (количество итераций меньше заданного числа или функция потерь велика), необходимо обучить НС. Процесс обучения заключается в изменении параметров НС (весов), т.е. выполняется подбор оптимальных значений весов с учётом функции потерь и обучаемой выборки (заданных ожидаемых значений). Для этого используется метод градиентного спуска и реализуется обратный проход.

Весы можно изменять в ручную, изменяя Tensors, содержащие обучаемые параметры (например с помощью `torch.no_grad()` или `.data`). Это удобно в случае простых алгоритмов оптимизации, таких как стохастический градиентный спуск, но на практике мы часто обучаем нейронные сети, используя более сложные методы AdaGrad, RMSProp, Adam и т. д. Пакет `optim` в PyTorch абстрагирует идею алгоритма оптимизации и предоставляет реализации часто используемых алгоритмов оптимизации (Таблица 17).

Таблица 17 – Пример оптимизаторов (подробнее см. <https://pytorch.org/docs/stable/optim.html>)

Название	Метод
SGD	стохастический градиентный спуск
Adam	адаптивная оценка моментов
RMSprop	алгоритм Джеффри Хинтона
LBFGS	алгоритм Бройдена-Флетчера-Гольдфарба-Шанно с ограниченным использованием памяти

Для использования оптимизатора необходимо на каждой итерации необходимо обнулять градиент, вызывать функцию `backward` и выполнять шаг оптимизатора.

Пример:

```
optimizer.zero_grad()
loss.backward()
optimizer.step()
```

Использование нейронной сети

После обучения НС ее требуется проверить на тестовой выборке. Если результат удовлетворителен, ее можно использовать для решения поставленной задачи, подавая на вход произвольные значения.

Примеры реализации нейронной сети

- 1) *Пример НС прямого распространения с функции активации ReLU, случайной выборкой, функцией потерь MSELoss, без использования оптимизатора, задана сеть с помощью nn.Sequential.*

Код:

```
# импорт библиотеки PyTorch
import torch

# задаем значения размерности
N, D_in, H, D_out = 64, 1000, 100, 10

# задаем случайным образом выборки
x = torch.randn(N, D_in)
y = torch.randn(N, D_out)

# строим модель НС
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
)

# выбрали функцию потерь
loss_fn = torch.nn.MSELoss(reduction='sum')

# скорость обучения
learning_rate = 1e-4

# цикл обучения с 500 эпохами
for t in range(500):
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    if t % 100 == 99:
        print(t, loss.item())
    model.zero_grad()
    loss.backward()

# расчет вручную параметрой НС
with torch.no_grad():
    for param in model.parameters():
        param -= learning_rate * param.grad
```

- 2) *Пример НС прямого распространения со случайной выборкой с 2 слоями, первый с функции активации ReLU, выходной с сигмоидальной функцией, используется для обучения метод обратного распространения и стохастический градиентный*

спуск (SGD), в качестве функции потерь средняя квадратичная функция (MSELoss).

Код:

```
# импорт библиотеки PyTorch
import torch
import torch.nn as nn

# определить все слои и размер пакета
# n_in - входной, n_h - скрытый, n_out - выходной, batch_size - пакет
обучающей выборки
n_in, n_h, n_out, batch_size = 10, 5, 1, 9

# заполняем случайными числами
# Возвращает тензор, заполненный случайными числами из нормального
распределения со средним 0 и дисперсией 1
# (также называемый стандартным нормальным распределением). Форма тензора
определяется переменным размером аргумента.

# входные данные
x = torch.randn(batch_size, n_in)
print(x)

# Создает тензор с данными.
# выходные данные
y = torch.tensor([[1.0], [0.0], [0.0], [1.0], [1.0], [1.0], [0.0], [0.0],
[1.0]])
print(y)

# class torch.nn.Sequential(*args) - Последовательный контейнер (модель НС).
# Модули будут добавлены к нему в порядке их передачи в конструктор.
# Линейное преобразование входных данных  $y=x*(A)T+b$ 
# определяем слой входов 10, слой с функцией ReLU() содержит 5 нейронов,
выходной нейрон 1 с функцией сигмоида
model = nn.Sequential(nn.Linear(n_in, n_h),
    nn.ReLU(),
    nn.Linear(n_h, n_out),
    nn.Sigmoid())

# выбрали функцию потерь MSELoss()
criterion = torch.nn.MSELoss()

# выбрали метод оптимизации и установили скорость обучения
optimizer = torch.optim.SGD(model.parameters(), lr = 0.01)

# модель градиентного спуска
```

```

# цикл обучения
for epoch in range(100):
    # Прямой проход: вычисляем выход НС, подав на вход модели начальные
    значения X
    y_pred = model(x)

    print(y_pred)
    # рассчитываем функцию потерь
    loss = criterion(y_pred, y)
    print('epoch: ', epoch, ' loss: ', loss.item())

    # Нулевые градиенты, выполнить обратный проход и обновить веса.
    # В PyTorch нам нужно установить градиенты на ноль, прежде чем начинать
    обратное распространение,
    # поскольку PyTorch накапливает градиенты при последующих обратных
    проходах.
    optimizer.zero_grad()

    # обратный проход (вычисляются градиенты)
    loss.backward()

    # шаг спуска градиента
    optimizer.step()

# новый входной вектор
x1 = torch.randn(1, n_in)
print(x1)

# получение выхода обученной НС
y_pred2 = model(x1)
print(y_pred2)

```

3) *Пример НС (Linear → ReLU → Linear), определённой с помощью nn.Module, с использованием оптимизатора Adam, с функцией потерь CrossEntropyLoss, обучающая выборка набор MNIST.*

Код:

```

import torch
import torch.nn as nn
import torchvision.datasets as datasets
import torchvision.transforms as transforms
from torch.autograd import Variable
# Размеры изображения = 28 x 28 = 784

```

```

input_size = 784
# Количество узлов на скрытом слое
hidden_size = 500
# Число классов на выходе. В этом случае от 0 до 9
num_classes = 10
# Количество тренировок всего набора данных
num_epochs = 5
# Размер входных данных для одной итерации
batch_size = 100
# Скорость обучения
learning_rate = 0.001
# Грузим набор данных MNIST
# обучающая выборка
train_dataset = datasets.MNIST(
    root='./data',
    train=True,
    transform=transforms.ToTensor(),
    download=True
)
# тестовая выборка
test_dataset = datasets.MNIST(
    root='./data',
    train=False,
    transform=transforms.ToTensor()
)
# создаем загрузчик для НС с перемешиванием для обучающей выборки и без
перемешивания для тестовой.
train_loader = torch.utils.data.DataLoader(
    dataset=train_dataset,
    batch_size=batch_size,
    shuffle=True
)
test_loader = torch.utils.data.DataLoader(
    dataset=test_dataset,
    batch_size=batch_size,
    shuffle=False
))
# Определяем вид НС
class Net(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(Net, self).__init__() # Наследуемый
родительским классом nn.Module

```

```

        self.fc1 = nn.Linear(input_size, hidden_size) # 1й связанный слой:
784 (данные входа) -> 500 (скрытый узел)
        self.relu = nn.ReLU() # Нелинейный слой ReLU
max(0,x)
        self.fc2 = nn.Linear(hidden_size, num_classes) # 2й связанный слой:
500 (скрытый узел) -> 10 (класс вывода)

    def forward(self, x): # Прямой проход
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        return out
# создаем объект НС
net = Net(input_size, hidden_size, num_classes)
# Определяем функцию потерь
criterion = nn.CrossEntropyLoss()
# Определяем оптимизатор
optimizer = torch.optim.Adam(net.parameters(), lr=learning_rate)
#Цикл обучения
for epoch in range(num_epochs):
# Грузим пакет изображений (index, data, class)
    for i, (images, labels) in enumerate(train_loader):
        # Преобразуем тензор в Variable: вектор 784 в матрицу 28 x 28
        images = Variable(images.view(-1, 28*28))
        labels = Variable(labels)
        # Инициализируем скрытые слои, веса=0
        optimizer.zero_grad()
        # Прямой проход по НС; вычисляем выход
        outputs = net(images)
        # Вычисляем функцию потерь (ошибку)
        loss = criterion(outputs, labels)
        # Обратный проход: корректировка весов
        loss.backward()
        # Выполняем шаг оптимизации (обучения)
        optimizer.step()
        # Выводим данные по обучению
        if (i+1) % 100 == 0:
            print('Epoch [%d/%d], Step [%d/%d], Loss: %.4f'%(epoch+1,
num_epochs, i+1, len(train_dataset)//batch_size, loss))
        correct = 0
        total = 0
    for images, labels in test_loader:
        images = Variable(images.view(-1, 28*28))

```

```

    outputs = net(images)
    _, predicted = torch.max(outputs.data, 1) # Выбор лучшего класса из
выходных данных: класс с лучшим счетом
    total += labels.size(0) # Увеличиваем суммарный счёт
    correct += (predicted == labels).sum() # Увеличиваем корректный счёт
print('Точность сети на 10К тестовых изображений: %d %%' % (100 * correct /
total))
torch.save(net.state_dict(), 'fnn_model.pkl')

```

4) *Пример Сверточной сети НС (Conv2d -> MaxPool2d -> Conv2d -> MaxPool2d -> Linear -> Linear), определённой с помощью nn.Module, с функцией потерь CrossEntropyLoss, случайной обучающей выборкой.*

Код:

```

import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F
class MNISTConvNet(nn.Module):
    def __init__(self):
        super(MNISTConvNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 10, 5)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(10, 20, 5)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(320, 50)
        self.fc2 = nn.Linear(50, 10)
    def forward(self, input):
        x = self.pool1(F.relu(self.conv1(input)))
        x = self.pool2(F.relu(self.conv2(x)))
        x = x.view(x.size(0), -1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return x
net = MNISTConvNet()
print(net)
input = Variable(torch.randn(1, 1, 28, 28))
out = net(input)
print(out.size())
target = Variable(torch.LongTensor([3]))
loss_fn = nn.CrossEntropyLoss()
err = loss_fn(out, target)
err.backward()
print(err)

```

```
print(net.conv1.weight.data.norm())
print(net.conv1.weight.grad.data.norm())
```

5) Пример рекуррентной сети с функцией `relu` обучаемой на входных данных набора MNIST, оптимизатором `SGD` и функцией потерь `CrossEntropyLoss`.

```
#https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_recurrent_neuralnetwork/
# подключаем библиотеки
import torch
import torch.nn as nn
import torchvision.transforms as transforms
import torchvision.datasets as dsets

# создаем обучающую и тестовую выборки на базе MNIST
train_dataset = dsets.MNIST(root='./data',
                             train=True,
                             transform=transforms.ToTensor(),
                             download=True)
test_dataset = dsets.MNIST(root='./data',
                            train=False,
                            transform=transforms.ToTensor())

# задаем параметры
batch_size = 100
n_iters = 3000
num_epochs = n_iters / (len(train_dataset) / batch_size)
num_epochs = int(num_epochs)
# подаем данные в загрузчик
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,
                                             batch_size=batch_size,
                                             shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset,
                                           batch_size=batch_size,
                                           shuffle=False)

# определяем рекуррентную нейронную сеть
class RNNModel(nn.Module):
    def __init__(self, input_dim, hidden_dim, layer_dim, output_dim):
        super(RNNModel, self).__init__()
        # размерность скрытых слоев]
        self.hidden_dim = hidden_dim
        # Количество скрытых слоев
        self.layer_dim = layer_dim
```

```

        self.rnn = nn.RNN(input_dim, hidden_dim, layer_dim, batch_first=True,
nonlinearity='relu')
        # Слой считывания
        self.fc = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        h0 = torch.zeros(self.layer_dim, x.size(0),
self.hidden_dim).requires_grad_()
        out, hn = self.rnn(x, h0.detach())
        # out.size() --> 100, 28, 10
        out = self.fc(out[:, -1, :])
        # out.size() --> 100, 10
        return out

# задаем параметры НС
input_dim = 28
hidden_dim = 100
layer_dim = 1
output_dim = 10
learning_rate = 0.01
criterion = nn.CrossEntropyLoss()
model = RNNModel(input_dim, hidden_dim, layer_dim, output_dim)
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
seq_dim = 28
iter = 0
for epoch in range(num_epochs):
    for i, (images, labels) in enumerate(train_loader):
        model.train()
        # Загрузка изображений в виде тензоров с возможностью накопления
градиента
        images = images.view(-1, seq_dim, input_dim).requires_grad_()
        # Обнуление градиента
        optimizer.zero_grad()

```

Варианты для выполнения лабораторной работы

Вариант	Функция потерь	Функция активации	НС	Оптимизатор	Входные данные
1	L1Loss	Sigmoid	Свёрточная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	MNIST
2	MSELoss	ReLU	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	SGD	CIFAR10

3	KLDivLoss	ELU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adam	STL10
4	BCELoss	Hardshrink	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	Adagrad	Рандом
5	BCEWithLogitsLoss	Hardtanh	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	CIFAR10
6	HingeEmbeddingLoss	LeakyReLU	Свёрточная НС (не менее 2 свёрточных слоёв и одного линейного)	Adam	STL10
7	KLDivLoss	LogSigmoid	Свёрточная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	CIFAR10
8	BCELoss	ELU	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	SGD	STL10
9	BCEWithLogitsLoss	Hardshrink	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adam	MNIST
10	L1Loss	Hardtanh	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	RMSProp	CIFAR10
11	MSELoss	LeakyReLU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	STL10
12	KLDivLoss	LogSigmoid	Свёрточная НС (не менее 2 свёрточных слоёв и одного линейного)	Adam	Рандом
13	BCELoss	Sigmoid	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	Adagrad	CIFAR10
14	KLDivLoss	LogSigmoid	Свёрточная НС (не менее 2 свёрточных слоёв и одного линейного)	RMSProp	CIFAR10
15	BCELoss	ELU	Рекуррентная нейронная сеть с 2 нелинейными слоями (readout)	SGD	STL10
16	BCEWithLogitsLoss	Hardshrink	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	Adam	CIFAR10
17		Hardtanh	НС прямого распространения (использовать не менее 2 разных нелинейных функций)	RMSProp	STL10

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

Список рекомендуемой литературы

основная

- 1) Смагин, А. А. Интеллектуальные информационные системы : учеб. пособие для вузов / А. А. Смагин, С. В. Липатова, А. С. Мельниченко ; УлГУ, Фак. математики и информ. технологий, Каф. телекоммуникац. технологий и сетей. - Ульяновск : УлГУ, 2010.- URL: <ftp://10.2.96.134/Text/smagin2.pdf>
- 2) Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для бакалавриата и магистратуры / Л. А. Станкевич. — Москва : Издательство Юрайт, 2019. — 397 с. — (Бакалавр и магистр. Академический курс). — ISBN 978-5-534-02126-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: <https://www.biblio-online.ru/bcode/433370>

дополнительная

- 3) Каку М., Будущее разума [Электронный ресурс] / Каку М. - М. : Альпина Паблишер, 2016. - 502 с. - ISBN 978-5-91671-369-5 - Режим доступа: <http://www.studentlibrary.ru/book/ISBN9785916713695.html>
- 4) Потопахин В.В., Романтика искусственного интеллекта / Потопахин В. В. - М. : ДМК Пресс, 2017. - 170 с. - ISBN 978-5-97060-476-2 - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : <https://www.studentlibrary.ru/book/ISBN9785970604762.html> (дата обращения: 10.11.2020). - Режим доступа : по подписке.
- 5) Цуканова Н.И., Онтологическая модель представления и организации знаний : Учебное пособие для вузов / Цуканова Н.И. - М. : Горячая линия - Телеком, 2015. - 272 с. - ISBN 978-5-9912-0454-5 - Текст : электронный // ЭБС "Консультант студента": [сайт]. - URL: <https://www.studentlibrary.ru/book/ISBN9785991204545.html> (дата обращения: 10.11.2020). - Режим доступа : по подписке.
- 6) Седова, Н. А. Теория нечетких множеств : учебное пособие / Н. А. Седова, В. А. Седов. — Саратов : Ай Пи Ар Медиа, 2019. — 421 с. — ISBN 978-5-4497-0196-1. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/86526.html>

- 7) Павлова, А. И. Информационные технологии: основные положения теории искусственных нейронных сетей : учебное пособие / А. И. Павлова. — Новосибирск : Новосибирский государственный университет экономики и управления «НИНХ», 2017. — 191 с. — ISBN 978-5-7014-0801-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/87110.html>

учебно-методическая

- 8) Седов, В. А. Введение в нейронные сети : методические указания к лабораторным работам по дисциплине «Нейроинформатика» для студентов специальности 09.03.02 «Информационные системы и технологии» / В. А. Седов, Н. А. Седова. — Саратов : Ай Пи Эр Медиа, 2018. — 30 с. — ISBN 978-5-4486-0047-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: <http://www.iprbookshop.ru/69319.html>

Программное обеспечение

1. Редактор онтологий Protégé (плагины SWRLTab, Pellet).
2. Anaconda (дистрибутив языков программирования Python и R), библиотеки (open source).