Министерство науки и высшего образования РФ Ульяновский государственный университет	Форма	
Ф-Рабочая программа дисциплины		

УТВЕРЖДЕНО

решением Учёного совета факультета математики,
информационных и авиационных технологий

от «21» июня 2019 г., протокол № 5/19

Председатель _____/ М.А. Волков
«21» июня 2019 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Дисциплина	Интеллектуальные информационные системы
Факультет	Факультет математики, информационных и авиационных технологий
Кафедра	Телекоммуникационные технологии и сети
Курс	4

Направление (специальность) 09.03.03 - " Прикладная информатика "
код направления (специальности), полное наименование

Направленность (профиль/специализация) Информационная сфера
полное наименование

Форма обучения очная
очная, заочная, очно-заочная (указать только те, которые реализуются)

Дата введения в учебный процесс УлГУ: «01» сентября 2017 г.

Программа актуализирована на заседании кафедры: протокол №8а__ от 11.03 2020_г.
Программа актуализирована на заседании кафедры: протокол №1___ от 31.08 2020_г.
Программа актуализирована на заседании кафедры: протокол №__ ____ от ___ __ 20_____г.

Сведения о разработчиках:

ФИО	Кафедра	Должность, ученая степень, звание
Липатова Светлана Валерьевна	Телекоммуникационных технологий и сетей	доцент, к.т.н., доцент

СОГЛАСОВАНО	СОГЛАСОВАНО
Заведующий реализующей кафедрой Телекоммуникационных технологий и сетей	Заведующий выпускающей кафедрой Информационных технологий
_____/ А.А.Смагин / *Подпись* *расшифровка подписи* «19» 06 2019 г.	(_____/ М.А.Волков/ *Подпись* *расшифровка подписи* «19» 06 2019 г.

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ:

Цели освоения дисциплины: формирование общекультурных и профессиональных компетенций, необходимых для реализации информационно-аналитической и научно-исследовательской деятельности

Задачи освоения дисциплины: приобретение в рамках освоения предусмотренного курсом занятий следующих знаний, умений и навыков, характеризующих определённый уровень сформированности целевых компетенций (см. подробнее п.3):

- сформировать системное базовое представление, первичные знания, умения и навыки студентов по основам инженерии знаний и нейроинформатики,
- дать общие представления о прикладных системах искусственного интеллекта,
- дать представление о роли искусственного интеллекта и нейроинфоратики в развитии информатики в целом, а также, в научно-техническом прогрессе,
- подготовить студентов к применению концепций интеллектуальных систем при дальнейшем обучении.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП:

Дисциплина «Системы искусственного интеллекта» относится к числу дисциплин вариативной части блока Б1.В.ОД, предназначенного для студентов, обучающихся по направлению: 09.03.03 "Прикладная информатика".

Для успешного изучения дисциплины необходимы знания и умения, приобретенные в результате освоения курсов «Дискретная математика и математическая логика», «Информатика и программирование», полностью или частично сформированные компетенции ОПК-1, ОПК-2, ОПК-3, УК-1, а именно:

- **знать:** основные понятия, утверждения, а так же методы исследования, методику построения различных дискретных структур, новейшие достижения дискретной математики, основные принципы программирования;
- **уметь:** применять методы дискретной математики на практике, работать в средах программирования;
- **владеть:** методологией и навыками решения научных и практических задач, писать программы на языках высокого уровня.

Основные положения дисциплины используются в дальнейшем при изучении дисциплины «Преддипломная практика».

3. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ), СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОСНОВНОЙ ПРОФЕССИОНАЛЬНОЙ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

Код и наименование реализуемой компетенции	Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций
ОПК-2 способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении	**знать:** – о истории, целях и задачах исследований в области искусственного интеллекта, – об областях применения интеллектуальных систем, – основные понятия нечетких вычислений, – об основных направлениях в исследованиях новых архитектур компьютеров,

задач профессиональной деятельности	– об основных этапах развития робототехники, – понятия инженерии знаний и нейрокибернетики, – прикладных системах искусственного интеллекта, **уметь:** – свободное использование терминологии как на русском, так и на английском языке (название операторов языка программирования, заимствованной терминологии) **владеть:** – навыками использования систем разработки интеллектуальных систем
ПК-2 способность разрабатывать и адаптировать прикладное программное обеспечение	**знать:** – о нечеткости знаний, ее природе и разновидностях, – основные моделей нейронных сетей, методы и алгоритмов их обучения, **уметь:** – ориентироваться в различных типах интеллектуальных систем, **владеть:** – методами представления и обработки знаний, – навыками формализации знаний экспертов с применением различных методов представления знаний,
ПК-3 способность проектировать ИС по видам обеспечения	**знать:** – этапы построения экспертных систем, – языках программирования искусственного интеллекта; – о принципах использования генетических алгоритмов, – понятия генетических алгоритмов, – о структуру экспертных систем и их архитектурные особенностей в зависимости от особенностей решаемой задачи, – о проблемах и способах построения нейронных сетей, **уметь:** – ориентироваться в различных методах представления знаний, **владеть:** – навыками использования нейронных сетей, эволюционных методов; – навыками нечеткого моделирования.
ПК-7 способность настраивать, эксплуатировать и сопровождать информационные системы и сервисы	**знать:** – о двух подходах к построению интеллектуальных систем – логическом и нейрокибернетическом, эволюционном, **уметь:** – ставить задачу построения экспертной системы для решения задачи выбора вариантов в плохо формализуемой предметной области,

	владеть: – навыками разработки онтологий; – навыками логического программирования;
ПК-9 способность осуществлять ведение базы данных и поддержку информационного обеспечения решения прикладных задач	**знать:** – проблемах и основных методах представления и обработки знаний, **уметь:** – осуществлять анализ предметной области, структурировать и формализовывать знания экспертной и их опыт; **владеть:** – навыками разработки продукционные базы знаний для решения задач задачи выбора вариантов в плохо формализуемой предметной области,

4. ОБЩАЯ ТРУДОЕМКОСТЬ ДИСЦИПЛИНЫ

4.1. Объем дисциплины в зачётных единицах (всего): 5

4.2. Объем дисциплины по видам учебной работы (в часах)

Вид учебной работы	**Количество часов (форма обучения очная)**	
	Всего по плану	**В т.ч. по семестрам**
		7 семестра
1	2	5
Контактная работа обучающихся с преподавателем	90	90
Аудиторные занятия:	90	90
Лекции	36	36
практические и семинарские занятия	18	18
лабораторные работы (лабораторный практикум)	36	36
Самостоятельная работа	90	90
Форма текущего контроля знаний и контроля самостоятельной работы	тестирование, контрольная работа (решений задач) 36	тестирование, контрольная работа (решений задач) 36
Курсовая работа	-	-
Виды промежуточной аттестации (экзамен, зачет)	экзамен (36)	экзамен (36)
Всего часов по дисциплине	216	216

4.3. Содержание дисциплины (модуля.) Распределение часов по темам и видам учебной работы:

Форма обучения очная

Название разделов и тем	Всего	Виды учебных занятий					Форма текущего контроля знаний
		Аудиторные занятия			Занятия в интерактивной форме	Самостоятельная работа	
		Лекции	Практические занятия, семинары	Лабораторные работы, практикумы			
1	2	3	4	5	6	7	8
1. Философские вопросы ИИ	12	2	2	-	-	8	Тестирование 5
2. Подходы и направления исследований в ИИ	20	6	2	-	-	12	Тестирование 5
3. Представление знаний	22	6	2	-	3	14	Проверка решения задач 5
4. Онтологии и Semantic Web	31	6	2	9	3	14	Тестирование 5
5. Эволюционное моделирование	31	6	2	9	4	14	Проверка решения задач 5
6. Нечеткие вычисления.	31	6	2	9	4	14	Проверка решения задач 5
7. Нейронные сети.	33	4	6	9	4	14	Проверка решения задач 6
Экзамен	36						
Итого	216	36	18	36	18*	90	36

5. СОДЕРЖАНИЕ ДИСЦИЛИНЫ (МОДУЛЯ)

Тема 1. Философские вопросы ИИ.

Определения естественного и искусственного интеллекта. Тест Тьюринга. Мысленный эксперимент «Китайская комната». Теорема Геделя. Технологическая сингулярность. Цели науки ИИ. Научная этика в ИИ.

Тема 2. Подходы и направления исследований в ИИ.

Предыстория, история развития искусственного интеллекта как научного

направления. Нейрокибернетика и кибернетика «черного ящика». История развития искусственного интеллекта в России. Основные направления исследований в области искусственного интеллекта. Свойства интеллектуальных информационных систем.

Тема 3. Представление знаний.

Данные и знания, свойства и виды знаний. Модели преставления знаний: логическая, фреймовая, продукционная и семантические сети. Их достоинства, недостатки, основные принципы построения и область использования.

Тема 4. Онтологии и Semantic Web.

Определение онтологий. Стек протоколов Semantic Web. Основные элементы онтологий: классы, индивиды, свойства, аксиомы. Запросы к онтологиям. Использование правил в онтологиях. Семантические машины вывода.

Тема 5. Эволюционное моделирование.

Эволюционное моделирование. Определение и основные понятия генетического алгоритма. Операторы кроссовер, мутация, и инверсия. Фито-функция. Методы отбора особей. Виды генетического алгоритма. Задачи, решаемые генетическим алгоритмом.

Тема 6. Нечеткие вычисления.

Теория нечётких множеств. Понятие нечеткого множества. Функция принадлежности. Операции с нечеткими множествами. Нечеткие отношения. Лингвистическая переменная. Нечеткие высказывания. Нечеткая импликация.

Тема 7. Нейронные сети.

Понятие нейрона. Модель математического нейрона. Персептрон Розенблатта. Правила Хебба. Алгоритм обучения по дельта-правилу. Проблема «исключающего или». Обучение многослойной нейронной сети методом обратного распространения ошибки. Классификация нейронных сетей. Задачи, решаемые нейронными сетями. Глубинное обучение. Свёрточные нейронные сети. Рекуррентные нейронные сети. Автокодировщик. Ограниченная машина Больцмана. Инициализация весов НС. Нормализация.

6. ТЕМЫ ПРАКТИЧЕСКИХ И СЕМИНАРСКИХ ЗАНЯТИЙ

Тема 1. Философские вопросы ИИ.

1) Какие угрозы видят ученые в развитии технологий ИИ?
2) Что такое технологическая сингулярность?
3) В чем заключался тест Тьюинга?
4) Какие задачи стоять перед современной наукой в области ИИ?
5) Как интерпретируют теорему Геделя по отношению к ИИ?
6) Как определяют предмет исследований науки об ИИ?
7) Какие задачи являются интеллектуальными?

Тема 2. Подходы и направления исследований в ИИ.

1) Какие задачи решает компьютерная лингвистика?
2) Какие виды анализа выполняют системы при машинном переводе?
3) Какие поколения роботов существуют?
4) Охарактеризуйте японский проект компьютер 5-го поколения?
5) На какие направления исследований делится искусственная жизнь?
6) Приведите примеры компьютерных игр с элементами ИИ и какие методы ИИ в них использовались?
7) Чем отличается слабый ИИ от сильного?

Тема 3. Представление знаний.

1) Достоинства и недостатки продукционной, семантической и фреймовой моделей?
2) Что из себя представляет Rete-алгоритм?

3) Что такое база знаний?
4) Примеры слабоструктурированных предметных областей?
5) Какие свойства отличают знания от данных?
6) Какого типа знания бывает?
7) Виды моделей представления знаний?

Тема 4. Онтологии и Semantic Web.

1) Для чего предназначена онтология?
2) Как соотносятся протоколы Semantic Web с семиуровневой моделью OSI?
3) Чем отличается Data Properties от Object Properties?
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?

Тема 5. Эволюционное моделирование.

1) Достоинства и недостатки эволюционных методов?
2) Какие существуют операторы ГА?
3) Приведите примеры оператора кроссовера?
4) Приведите примеры оператора мутации?
5) Что из себя представляет островная модель ГА?
6) Как выполняется турнирный отбор?
7) Что подразумевает принцип элитизма?

Тема 6. Нечеткие вычисления.

1) Как задается нечеткое множество?
2) Кае значения может принимать функция принадлежности?
3) Приведите пример нечеткой переменной.
4) Приведите пример лингвистической переменной?
5) Как проверяется полнота нечеткой базы знаний?
6) Что такое фаззицикация и дефаззификация?
7) Какие методы применяются на этапе аккумуляции?

Тема 7. Нейронные сети.

1) Какие достоинства и недостатки у НС?
2) Как инициализируют синаптические веса НС?
3) Какие задачи можно решать с помощью НС?
4) Что из себя представляет пакетная нормализация?
5) Как выполняется операция свёртка в сверточной НС?
6) Как используется градиент в обучении НС?
7) Какие нейронные сети относятся к глубинным?

Для практических заданий использовать:

Сборник задач по курсу "Интеллектуальные информационные системы" : учеб. пособие для вузов / С. В. Липатова; УлГУ, Фак. математики и информ. технологий", Каф. телекоммуникац. технологий и сетей. - Ульяновск : УлГУ, 2010. - 64 с. : ил. - Библиогр.: с. 64. - б/п.

7. ЛАБОРАТОРНЫЕ РАБОТЫ, ПРАКТИКУМЫ

Тема 4. Онтологии и Semantic Web.

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 антологии.

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.

Основные теоретические сведения

В последние годы разработка онтологий - формальных явных описаний терминов предметной области и отношений между ними (Gruber 1993) – переходит из мира лабораторий по искусственному интеллекту на рабочие столы экспертов по предметным областям. Во всемирной паутине онтологии стали обычным явлением. Онтологии в сети варьируются от больших таксономий, категоризирующих веб-сайты (как на сайте Yahoo!), до категоризаций продаваемых товаров и их характеристик (как на сайте Amazon.com). Консорциум WWW (W3C) разрабатывает RDF (Resource Description Framework) (Brickley and Guha 1999), язык кодирования знаний на веб-страницах, для того, чтобы сделать их понятными для электронных агентов, которые осуществляют поиск информации. Управление перспективных исследований и разработок министерства обороны США (The Defense Advanced Research Projects Agency, DARPA) в сотрудничестве с W3C разрабатывает Язык Разметки для Агентов DARPA (DARPA Agent Markup Language, DAML), расширяя RDF более выразительными конструкциями, предназначенными для облегчения взаимодействия агентов в сети (Hendler and McGuinness 2000).

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**)).

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.

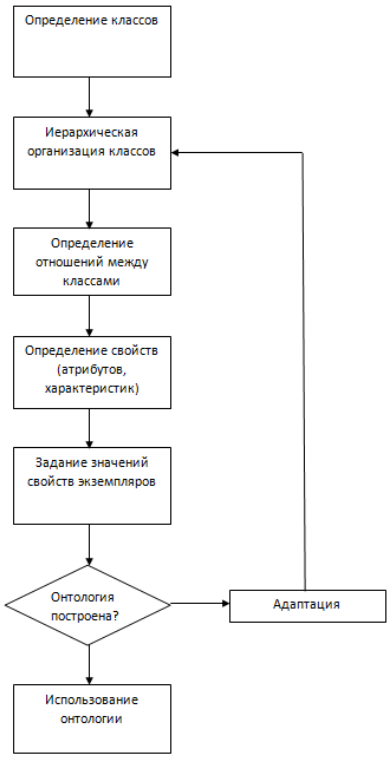
Слоты описывают свойства классов и экземпляров: вино крепкое, оно производится на винном заводе Абрау-Дюрсо. У нас есть два слота, которые описывают вино в этом примере: слот крепость со значением «крепкое» и слот производитель со значением «винный завод Абрау-Дюрсо. Мы можем сказать, что на уровне класса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

Схема процесса создания онтологии



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 a и индивида b, а также индивида b связывает с индивидом c, то мы можем вывести, что индивид a связан с индивидом c через это свойство.

Симметричные свойства (Simmetric) – если свойство p симметричное, и свойство связывает индивида a с индивидом b, то индивид b связан также с индивидом a через свойство p.

Асимметричные свойства (Asimmetric) – если свойство p асимметричное, и свойство связывает индивида a с индивидом b, то индивид b не может быть связан с индивидом a через свойство p.

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

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

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 This is the most common type of class expression. Also known as, an "SomeValueFrom restriction" or an "Existential Restriction". This kind of class expression consists of a property (in this case hasPet) and a class expression that is known as a filler (in this case the filler is Dog). Individuals that are instances of this class expression have a relationship along the hasPet property to an individual that is an instance of class Dog.
value	hasPet value Tibbs	Things that have a pet that is Tibbs. Here, Tibbs is a specific individual. Intuitively this would describe Tibb's owners. Note that this is a short cut for, and logically equivalent to, (hasPet some {Tibbs}), where the curly brackets describe a class of specific individuals - in this case, a class of one individual that is Tibbs. Also known as a "HasValue restriction"
only	hasPet only Cat	Things that have pets that are only Cats. Note that this does not mean that these things must have a Cat, but if they do have a pet then it will be a Cat. Also known as an "AllValuesFrom restriction" or a "Universal restriction"
min	hasPet min 3 Cat	Things that have at least three pets that are Cats. Things that have at least three pets that are Cats. Also known as a "Min cardinality restriction"
max	hasPet max 5 Dog	Things that have at most five pets that are Dogs. Note that there may be more than five pets in total e.g. three Cats and five Dogs, but the number of Dogs will not be more than five. This class expression also means that there may be no pets at all. Also known as a "Max cardinality restriction"
exactly	hasPet exactly 2 GoldFish	Things that have exactly 2 GoldFish as pets. Note that there may be more than two pets, but two of the pets are GoldFish - no more and no less. Also known as an "Exact cardinality restriction"
and	Person and (hasPet some Cat)	People that have a pet that's a Cat. Here we have a class name and a complex class expression combined with the and keyword. The brackets are optional in this particular case, but have been included for clarity. Also known as an "Intersection" or a "Conjunction". We also say that each class expression in the conjunction is a "conjunct".
or	(hasPet some Cat) or (hasPet	Things that have a pet that's a Cat or have a pet that's a Dog Note that the or is not exclusive. That is, this class includes the things that have a Cat

	some Dog)	as a pet, or have a Dog as a pet, or have both a pet Dot and a pet Cat. Note that we could have also written this example as (hasPet some (Cat or Dog)) Also known as a "Union" or a "Disjunction". We also say that each class expression in the disjunction is a "disjunct".
not	not (hasPet some Dog)	Things that do not have a pet that's a dog. This includes things that either do not have any pet, or if they have a pet then it's not a Dog. Note that this is logically equivalent to (hasPet only (not Dog)) Also known as a "negation".

Список команд Built-Ins (можно применять в правилах)

Команда	Описание
for Comparisons	
swrlb:equal (from XQuery op:numeric-equal, op:compare, op:boolean-equal op:yearMonthDuration-equal, op:dayTimeDuration-equal, op:dateTime-equal, op:date-equal, op:time-equal, op:gYearMonth-equal, op:gYear-equal, op:gMonthDay-equal, op:gMonth-equal, op:gDay-equal, op:anyURI-equal)	Satisfied iff the first argument and the second argument are the same.
swrlb:notEqual (from swrlb:equal)	The negation of swrlb:equal.
swrlb:lessThan (from XQuery op:numeric-less-than, op:compare, op:yearMonthDuration-less-than, op:dayTimeDuration-less-than, op:dateTime-less-than, op:date-less-than, op:time-less-than)	Satisfied iff the first argument and the second argument are both in some implemented type and the first argument is less than the second argument according to a type-specific ordering (partial or total), if there is one defined for the type. The ordering function for the type of untyped literals is the partial order defined as string ordering when the language tags are the same (or both missing) and incomparable otherwise.
swrlb:lessThanOrEqual (from swrlb:lessThan, swrlb:equal)	Either less than, as above, or equal, as above.
swrlb:greaterThan (from XQuery op:numeric-greater-than, op:compare, op:yearMonthDuration-greater-than, op:dayTimeDuration-greater-than, op:dateTime-greater-than, op:date-greater-than, op:time-greater-than)	Similarly to swrlb:lessThan
swrlb:greaterThanOrEqual (from swrlb:greaterThan, swrlb:equal)	Similarly to swrlb:lessThanOrEqual.
Math	
swrlb:add (from XQuery op:numeric-add)	Satisfied iff the first argument is equal to the arithmetic sum of the second argument through the last argument.
swrlb:subtract (from XQuery op:numeric-subtract)	Satisfied iff the first argument is equal to the arithmetic difference of the second argument minus the third argument.
swrlb:multiply (from XQuery op:numeric-multiply)	Satisfied iff the first argument is equal to the arithmetic product of the second argument through the last argument.
swrlb:divide (from XQuery op:numeric-divide)	Satisfied iff the first argument is equal to the arithmetic quotient of the second argument divided by the third argument.
swrlb:integerDivide (from XQuery op:numeric-integer-divide)	Satisfied if the first argument is the arithmetic quotient of the second argument idiv the third argument. If the numerator is not evenly divided by the divisor, then the quotient is the xsd:integer value obtained, ignoring any remainder that results from the division (that is, no rounding is performed).
swrlb:mod (from XQuery op:numeric-mod)	Satisfied if the first argument represents the remainder resulting

	from dividing the second argument, the dividend, by the third argument, the divisor. The operation a mod b for operands that are xsd:integer or xsd:decimal, or types derived from them, produces a result such that (a idiv b)*b+(a mod b) is equal to a and the magnitude of the result is always less than the magnitude of b. This identity holds even in the special case that the dividend is the negative integer of largest possible magnitude for its type and the divisor is -1 (the remainder is 0). It follows from this rule that the sign of the result is the sign of the dividend
swrlb:pow	Satisfied iff the first argument is equal to the result of the second argument raised to the third argument power.
swrlb:unaryPlus (from XQuery op:numeric-unary-plus)	Satisfied iff the first argument is equal to the second argument with its sign unchanged.
swrlb:unaryMinus (from XQuery op:numeric-unary-minus)	Satisfied iff the first argument is equal to the second argument with its sign reversed.
swrlb:abs (from XQuery fn:abs)	Satisfied iff the first argument is the absolute value of the second argument.
swrlb:ceiling (from XQuery fn:ceiling)	Satisfied iff the first argument is the smallest number with no fractional part that is greater than or equal to the second argument.
swrlb:floor (from XQuery fn:floor)	Satisfied iff the first argument is the largest number with no fractional part that is less than or equal to the second argument
swrlb:round (from XQuery fn:round)	Satisfied iff the first argument is equal to the nearest number to the second argument with no fractional part.
swrlb:roundHalfToEven (from XQuery fn:round-half-to-even)	Satisfied iff the first argument is equal to the second argument rounded to the given precision. If the fractional part is exactly half, the result is the number whose least significant digit is even.
swrlb:sin	Satisfied iff the first argument is equal to the sine of the radian value the second argument.
swrlb:cos	Satisfied iff the first argument is equal to the cosine of the radian value the second argument.
swrlb:tan	Satisfied iff the first argument is equal to the tangent of the radian value the second argument.
for Boolean Values	
swrlb:booleanNot (from XQuery fn:not)	Satisfied iff the first argument is true and the second argument is false, or vice versa.
for Strings	
swrlb:stringEqualIgnoreCase	Satisfied iff the first argument is the same as the second argument (upper/lower case ignored)
swrlb:stringConcat (from XQuery fn:concat)	Satisfied iff the first argument is equal to the string resulting from the concatenation of the strings the second argument through the last argument.
swrlb:substring (from XQuery fn:substring)	Satisfied iff the first argument is equal to the substring of optional length the fourth argument starting at character offset the third argument in the string the second argument.
swrlb:stringLength (from XQuery fn:string-length)	Satisfied iff the first argument is equal to the length of the second argument.
swrlb:normalizeSpace (from XQuery fn:normalize-space)	Satisfied iff the first argument is equal to the whitespace-normalized value of the second argument.
swrlb:upperCase (from XQuery fn:upper-case)	Satisfied iff the first argument is equal to the upper-cased value of the second argument.
swrlb:lowerCase (from XQuery fn:lower-case)	Satisfied iff the first argument is equal to the lower-cased value of

	the second argument.
swrlb:translate (from XQuery fn:translate)	Satisfied iff the first argument is equal to the second argument with occurrences of characters contained in the third argument replaced by the character at the corresponding position in the string the fourth argument.
swrlb:contains (from XQuery fn:contains)	Satisfied iff the first argument contains the second argument (case sensitive).
swrlb:containsIgnoreCase	Satisfied iff the first argument contains the second argument (case ignored).
swrlb:startsWith (from XQuery fn:starts-with)	Satisfied iff the first argument starts with the second argument.
swrlb:endsWith (from XQuery fn:ends-with)	Satisfied iff the first argument ends with the second argument.
swrlb:substringBefore (from XQuery fn:substring-before)	Satisfied iff the first argument is the characters of the second argument that precede the characters of the third argument.
swrlb:substringAfter (from XQuery fn:substring-after)	Satisfied iff the first argument is the characters of the second argument that follow the characters of the third argument.
swrlb:matches (from XQuery fn:matches)	Satisfied iff the first argument matches the regular expression the second argument.
swrlb:replace (from XQuery fn:replace)	Satisfied iff the first argument is equal to the value of the second argument with every substring matched by the regular expression the third argument replaced by the replacement string the fourth argument.
swrlb:tokenize (from XQuery fn:tokenize)	Satisfied iff the first argument is a sequence of one or more strings whose values are substrings of the second argument separated by substrings that match the regular expression the third argument.
for Date, Time and Duration	
swrlb:yearMonthDuration (from XQuery xdt:yearMonthDuration)	Satisfied iff the first argument is the xsd:duration representation consisting of the year the second argument and month the third argument.
swrlb:dayTimeDuration (from XQuery xdt:dayTimeDuration)	Satisfied iff the first argument is the xsd:duration representation consisting of the days the second argument, hours the third argument, minutes the fourth argument, and seconds the fifth argument.
swrlb:dateTime	Satisfied iff the first argument is the xsd:dateTime representation consisting of the year the second argument, month the third argument, day the fourth argument, hours the fifth argument, minutes the sixth argument, seconds the seventh argument, and timezone the eighth argument.
swrlb:date	Satisfied iff the first argument is the xsd:date representation consisting of the year the second argument, month the third argument, day the fourth argument, and timezone the fifth argument.
swrlb:time	Satisfied iff the first argument is the xsd:time representation consisting of the hours the second argument, minutes the third argument, seconds the fourth argument, and timezone the fifth argument.
swrlb:addYearMonthDurations (from XQuery op:add-yearMonthDurations)	Satisfied iff the yearMonthDuration the first argument is equal to the arithmetic sum of the yearMonthDuration the second argument through the yearMonthDuration the last argument.
swrlb:subtractYearMonthDurations (from XQuery op:subtract-yearMonthDurations)	Satisfied iff the yearMonthDuration the first argument is equal to the arithmetic difference of the yearMonthDuration the second argument minus the yearMonthDuration the third argument.

swrlb:multiplyYearMonthDuration (from XQuery op:multiply-yearMonthDuration)	Satisfied iff the yearMonthDuration the first argument is equal to the arithmetic product of the yearMonthDuration the second argument multiplied by the third argument.
swrlb:divideYearMonthDurations (from XQuery op:divide-yearMonthDuration)	Satisfied iff the yearMonthDuration the first argument is equal to the arithmetic remainder of the yearMonthDuration the second argument divided by the third argument.
swrlb:addDayTimeDurations (from XQuery op:add-dayTimeDurations)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic sum of the dayTimeDuration the second argument through the dayTimeDuration the last argument.
swrlb:subtractDayTimeDurations (from XQuery op:subtract-dayTimeDurations)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic difference of the dayTimeDuration the second argument minus the dayTimeDuration the third argument.
swrlb:multiplyDayTimeDurations (from XQuery op:multiply-dayTimeDuration)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic product of the dayTimeDuration the second argument multiplied by the third argument.
swrlb:divideDayTimeDuration (from XQuery op:divide-dayTimeDuration)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic remainder of the dayTimeDuration the second argument divided by the third argument.
swrlb:subtractDates (from XQuery op:subtract-dates)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic difference of the xsd:date the second argument minus the xsd:date the third argument.
swrlb:subtractTimes (from XQuery op:subtract-times)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic difference of the xsd:time the second argument minus the xsd:time the third argument.
swrlb:addYearMonthDurationToDateTime (from XQuery op:add-yearMonthDuration-to-dateTime)	Satisfied iff the xsd:dateTime the first argument is equal to the arithmetic sum of the xsd:dateTime the second argument plus the yearMonthDuration the third argument.
swrlb:addDayTimeDurationToDateTime (from XQuery op:add-dayTimeDuration-to-dateTime)	Satisfied iff the xsd:dateTime the first argument is equal to the arithmetic sum of the xsd:dateTime the second argument plus the dayTimeDuration the third argument.
swrlb:subtractYearMonthDurationFromDateTime (from XQuery op:subtract-yearMonthDuration-from-dateTime)	Satisfied iff the xsd:dateTime the first argument is equal to the arithmetic difference of the xsd:dateTime the second argument minus the yearMonthDuration the third argument.
swrlb:subtractDayTimeDurationFromDateTime (from XQuery op:subtract-dayTimeDuration-from-dateTime)	Satisfied iff the xsd:dateTime the first argument is equal to the arithmetic difference of the xsd:dateTime the second argument minus the dayTimeDuration the third argument.
swrlb:addYearMonthDurationToDate (from XQuery op:add-yearMonthDuration-to-date)	Satisfied iff the xsd:date the first argument is equal to the arithmetic sum of the xsd:date the second argument plus the yearMonthDuration the third argument.
swrlb:addDayTimeDurationToDate (from XQuery op:add-dayTimeDuration-to-date)	Satisfied iff the xsd:date the first argument is equal to the arithmetic sum of the xsd:date the second argument plus the dayTimeDuration the third argument.
swrlb:subtractYearMonthDurationFromDate (from XQuery op:subtract-yearMonthDuration-from-date)	Satisfied iff the xsd:date the first argument is equal to the arithmetic difference of the xsd:date the second argument minus the yearMonthDuration the third argument.
swrlb:subtractDayTimeDurationFromDate (from XQuery op:subtract-dayTimeDuration-from-date)	Satisfied iff the xsd:date the first argument is equal to the arithmetic difference of the xsd:date the second argument minus the yearMonthDuration the third argument.
swrlb:addDayTimeDurationToTime (from XQuery op:add-dayTimeDuration-to-time)	Satisfied iff the xsd:time the first argument is equal to the arithmetic sum of the xsd:time the second argument plus the dayTimeDuration the third argument.
swrlb:subtractDayTimeDurationFromTime (from XQuery op:subtract-dayTimeDuration-from-time)	Satisfied iff the xsd:time the first argument is equal to the arithmetic difference of the xsd:time the second argument minus

	the dayTimeDuration the third argument.
swrlb:subtractDateTimesYieldingYearMonthDuration (from XQuery fn:subtract-dateTimes-yielding-yearMonthDuration)	Satisfied iff the yearMonthDuration the first argument is equal to the arithmetic difference of the xsd:dateTime the second argument minus the xsd:dateTime the third argument.
swrlb:subtractDateTimesYieldingDayTimeDuration (from XQuery fn:subtract-dateTimes-yielding-dayTimeDuration)	Satisfied iff the dayTimeDuration the first argument is equal to the arithmetic difference of the xsd:dateTime the second argument minus the xsd:dateTime the third argument.
for URIs	
swrlb:resolveURI (from XQuery op:resolve-uri)	Satisfied iff the URI reference the first argument is equal to the value of the URI reference the second argument resolved relative to the base URI the third argument.
swrlb:anyURI	Satisfied iff the first argument is a URI reference consisting of the scheme the second argument, host the third argument, port the fourth argument, path the fifth argument, query the sixth argument, and fragment the seventh argument.
for Lists	
swrlb:listConcat (from Common Lisp append)	Satisfied iff the first argument is a list representing the concatenation of the lists the second argument through the last argument.
swrlb:listIntersection	Satisfied iff the first argument is a list containing elements found in both the list the second argument and the list the third argument.
swrlb:listSubtraction	Satisfied iff the first argument is a list containing the elements of the list the second argument that are not members of the list the third argument.
swrlb:member	Satisfied iff the first argument is a member of the list the second argument.
swrlb:length (from Common Lisp list-length)	Satisfied iff the first argument is the length of the list the second argument (number of members of the list).
swrlb:first (from rdf:first)	Satisfied iff the first argument is the first member of the list the second argument.
swrlb:rest (from rdf:rest)	Satisfied iff the first argument is a list containing all members of the list the second argument except the first member (the head).
swrlb:sublist	Satisfied iff the list the first argument contains the list the second argument.
swrlb:empty (from rdf:nil)	Satisfied iff the list the first argument is an empty list.

Варианты заданий

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 5. Эволюционное моделирование.

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:

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

Варианты

	Целевая функция	Вид особи	Приспособленность	Отбор	Кроссовер	Мутация	Стратегия
1	Максимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторятся, 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 повторятся, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Отбор рулеткой	Упорядоченный	Меняет на противоположное значение ген	(μ+λ)
3	Максимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, 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	(μ+λ)
5	Минимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторятся, длина особи: 10	Среднее значение генов особей популяции	Выбор лучших k особей	Равномерный	Гауссовская мутация	(μ+λ)
6	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторятся, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Выбор случайных k особей	Упорядоченный	Меняет на противоположное значение ген	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
7	Минимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, 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 значении генов	Турнирный отбор	Равномерный	Меняет на противоположное значение ген	(μ+λ)
9	Максимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторятся, 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 повторятся, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Выбор лучших k особей	Упорядоченный	Меняет на противоположное значение ген	(μ+λ)
11	Максимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, длина особи: 10	Среднее значение генов особей популяции	Выбор случайных k особей	Равномерный	Гауссовская мутация	(μ, λ)
12	Минимум,	Тип: булевый длина особи: 10	Количество элементов =1	Выбор	Одноточечн	Полиномиальна	Оба потомка заменяют

	однокритериальная		(истина) минус 5 ген	случайных k особей	ый	я мутация	родителей. Мутировать могут и родители и потомки.
13	Минимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторятся, длина особи: 10	Среднее значение генов особей популяции	Турнирный отбор	Одноточечный	Гауссовская мутация	(μ+λ)
14	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторятся, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Турнирный отбор	Упорядоченный	Меняет на противоположное значение ген	(μ+λ)
15	Минимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, 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	(μ+λ)
17	Максимум, однокритериальная	Тип: массив вещественных чисел (array), значения: от 0 до 1 могут повторятся, 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 повторятся, длина особи: 5	Сумма значений генов особи за исключением чётных генов	Выбор лучших k особей	Упорядоченный	Меняет на противоположное значение ген	(μ+λ)
19	Максимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, 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 (array), значения: от 0 до 1 могут повторятся, длина особи: 10	Среднее значение генов особей популяции	Отбор рулеткой	Одноточечный	Гауссовская мутация	(μ+λ)
22	Максимум, однокритериальная	Тип: список целых чисел, значения: от 0 до 10 не могут повторятся, длина особи: 5	Сумма значений генов особи за исключением нечётных генов	Выбор лучших k особей	Равномерный	Меняет на противоположное значение ген	(μ, λ)
23	Минимум, однокритериальная	Тип: массив вещественных чисел (numpy), значения: от 0 до 13 могут повторятся, длина особи: 10	Среднее значение генов особей популяции делённая на 2 ген особи	Выбор случайных k особей	Одноточечный	Гауссовская мутация	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.

Методические указания по использованию библиотеки DEAP для работы с генетическими алгоритмами

1.1 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 1, Таблица 1).

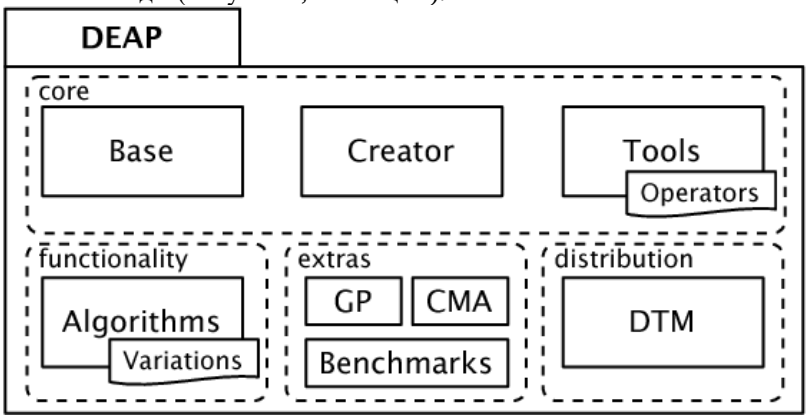


Рисунок 1 – Схема классов библиотеки[1]

Таблица 1 - Модули библиотеки для реализации ГА

Модуль	Описание
deap.creator	Содержит класс `deap.creator.create, необходим для создания новых классов` с именем name, наследующий классы из пакета base. Новый класс может иметь аразные трибуты.
deap.base	Пакет содержит базовые классы. 1) deap.base.Toolbox - набор инструментов для эволюционных методов (на основе классов пакета tools), 2) `deap.base.Fitness` – пригодность / приспособленность (показатель качества решения) особи. 3) *class* `deap.base.Tree (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 поддерево, 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 реалиации эволюционных методов, при этом используются операторы, зарегистрированные в соответсвующем объекте Toolbox. Обычно используются следующие ключевые слова: mate () для кроссовера, mutate () для мутации, select () для выбора.

1.2 Реализация генетического алгоритма

1.2.1 Определение приспособленности особей и вида особи (индивида)

Необходимо определить класс для приспособленности и особи (индивида), для этого нужно использовать

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

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 weights**
Однокритериальная на минимум	weights=(-1.0,)
Однокритериальная на максимум	weights=(1.0,)
Многокритериальная оптимизация (2 критерия)	weights=(-1.0, 1.0)

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) 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)
```

1.2.2 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

Инициали-зация	**Отбор**	**Скрещивание**	**Мутация**
initRepeat()	selTournament()	cxOnePoint()	mutGaussian()
initIterate()	selRoulette()	cxTwoPoint()	mutShuffleIndexes()
initCycle()	selNSGA2()	cxUniform()	mutFlipBit()
	selNSGA3()	cxPartialyMatched()	mutPolynomialBounded()
	selSPEA2()	cxUniformPartialyMatched()	mutUniformInt()
	selRandom()	cxOrdered()	mutESLogNormal()
	selBest()	cxBlend()	
	selWorst()	cxESBlend()	
	selTournamentDCD()	cxESTwoPoint()	
	selDoubleTournament()	cxSimulatedBinary()	
	selStochasticUniversalSampling()	cxSimulatedBinaryBounded()	
	selLexicase()	cxMessyOnePoint()	
	selEpsilonLexicase()		**Миграция**
	selAutomaticEpsilonLexicase()		migRing()

1.2.2.1 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 паз 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("attr_flt", random.uniform, 0.1, 0.7)
toolbox.register("individual", tools.initCycle, creator.Individual_1,
(toolbox.attr_int, toolbox.attr_flt), n=4)
```

1.2.2.2 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;*
- tournsize - количество участников тура (для турнирного отбора).

Таблица 5 – Пример операторов отбора для ГА

Функция	**Параметры**	**Вид**	**Описание**
`deap.tools.selTournament`	(*individuals, k, tournsize, 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
deap.tools.selRoulette	*(individuals, k, 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 $P_s(i)$)
`deap.tools.selRandom`	*(individuals, k)*	`Выбор случайных k особей`	`Случайная отбор`
`deap.tools.selBest`	(*individuals, k, fit_attr='fitness'*)	`Выбор лучших k особей`	`Отбор усечением`
`deap.tools.selWorst`	(*individuals, k, 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)
```

1.2.2.3 Скрещивание

В модуле tools представлена реализация 12 видов операторов кроссовера (Таблица 4). Рассмотрим подробнее 4 из них (

Таблица 6), 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()	(*ind1*, *ind2*)	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()	(*ind1*, *ind2*)	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`	(*ind1*, *ind2*, *indpb*)	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:** 100001100001 **Потомок 2:** 010101101111
`deap.tools.cxOrdered`	(*ind1*, *ind2*)	Упорядоченный	1) Выбор двух точек разрыва. 2) Обмен центральными	**Родитель 1:** 123|4567|89 **Родитель 2:** 375|2814|96

			частями 2) Обход всех незадействованных 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).	**Потомок 1:** 567 281493 **Потомок 2:** 281456793 9 – остался на месте, остальные сдвинулись

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

Пример:

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

1.2.2.4 Мутация

Существует много стратегий отбора, 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 больше 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**
deap.tools.mutGaussian	(*individual*, *mu*, *sigma*, *indpb*)¶	добавляет случайное число, заданное гауссовым распределением с нулевым средним для каждого компонента входного вектора (применяется для вещественного типа)
deap.tools.mutFlipBit	(*individual*, *indpb*)	Меняет на противоположное значение бит с определённой вероятностью (применяется к логическому типу)
deap.tools.mutPolynomialBounded	(*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)
```

1.2.2.5 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)
```

1.2.3 Определение инструментов контроля функционирования генетического алгоритма

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

Таблица 8 – Пример классов для хранения данных о функционировании ГА

Класс		Описание
`deap.tools.Statistics`	([*key*]) – 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 *register*
`deap.tools.Logbook`	нет	Объект - эволюционные записи в виде хронологического списка словарей. Данные могут быть получены с помощью метода select.
`deap.tools.HallOfFame`	(*maxsize*, *similar*=<*built-in function eq*>) *Maxsize* – максимальное количество особей в зале славы *similar* - Оператор эквивалентности между двумя особями (необязательный параметр)	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")
```

1.2.4 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 python (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**
deap.algorithms.eaSimple	(*population*, *toolbox*, *cxpb*, *mutpb*, *ngen*[, *stats*, *halloffame*, *verbose*])	Оба потомка заменяют родителей. Мутировать могут и родители и потомки.
deap.algorithms.eaMuPlusLambda	(population, toolbox, mu, lambda_, cxpb, mutpb, ngen[, stats, halloffame, verbose])	Только первый ребёнок добавляете в популяцию, второй отбрасывается. Потомок не подвергается мутации (мутируют родители). В (μ+λ)-стратегиях селекция производится из (μ+λ) особей - объединённой популяции родителей и потомков. Необходимо регистрировать функции: toolbox.mate(), toolbox.mutate(), toolbox.select() и

		toolbox.evaluate().
deap.algorithms.eaMuCommaLambda	(population, toolbox, mu, lambda_, cxpb, mutpb, ngen[, stats, halloffame, verbose])	Только первый ребёнок добавляете 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:

4) 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)
```

5) 2) Стратегия (μ+λ).

```
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)
```

1.2.5 Просмотр результатов функционирования генетического алгоритма

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

1.3 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)
    # расчитываем массив приспособленности
    fitnesses = list(map(toolbox.evaluate, pop))
```

```
    # присваиваем значение приспособленности соответсвующим особям
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
    fits = [ind.fitness.values[0] for ind in pop]
    #параметы 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 невутированный вариан 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 знаяение %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)
```

Тема 6. Нечеткие вычисления.

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:

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

Методические указания по использованию библиотеки Skfuzzy для построения нечётких систем

1.4 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 2).

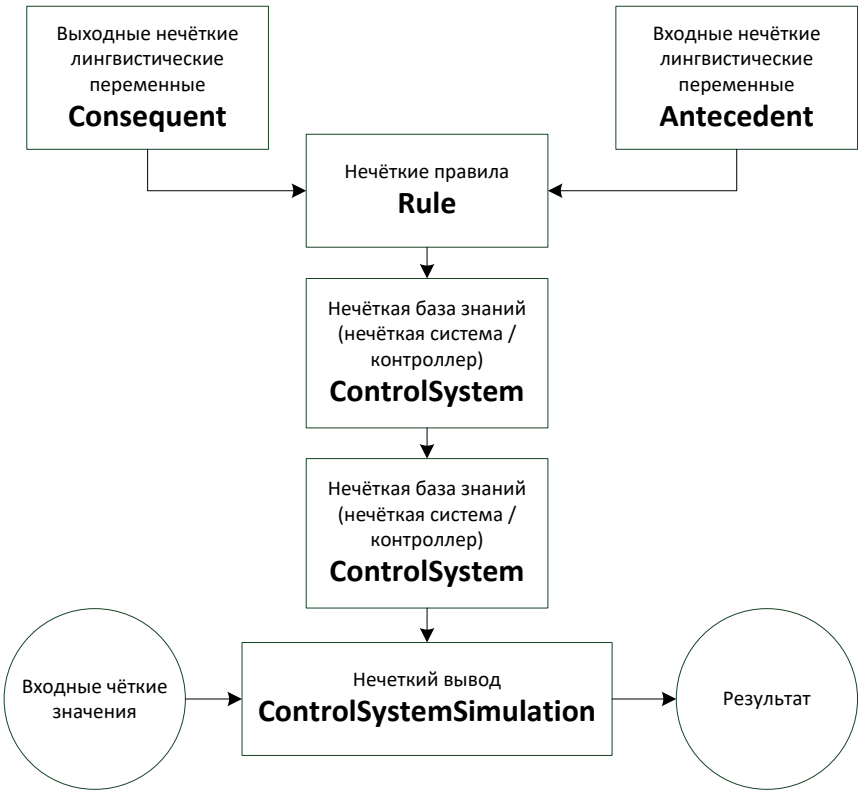


Рисунок 2 – Схема применения классов пакета Control

1.5 Определение лингвистических переменных

1.5.1 Определение лингвистической переменной

При описании лингвистической переменной необходимо определить входная она для задачи или выходная. Входная лингвистическая переменная является антецедентом (предшествующим условием), а выходная – консеквентом (следствием).
Для входной лингвистической переменной используется метод `skfuzzy.control.Antecedent`(*universe*, *label*), для выходной - `skfuzzy.control.Consequent`(*universe*, *label*), где *label – название переменной*, *universe – универсум (четкое множество, на котором задаётся нечёткая переменная), одномерный конвертируемый в NumPy массив.*
Сам массив можно определит разными способами (см. например https://pyprog.pro/introduction.html).

1.5.2 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).

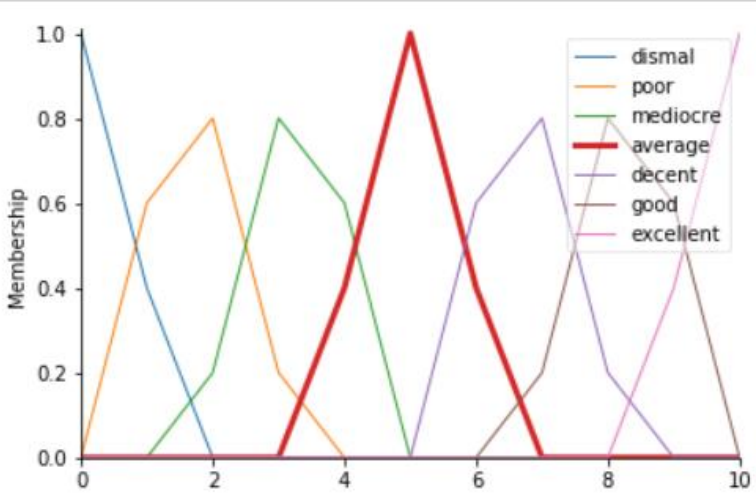
Таблица 11 – Примеры методы генерации функций принадлежности (см. подробнее ***https://pythonhosted.org/scikit-fuzzy/api/skfuzzy.membership.html****)*

Метод	**Описание**	**Формула**
skfuzzy.membership.dsigmf(x, b1, c1, b2, c2)	Разница двух нечётких сигмоидальных функций принадлежности.	y = f1 - f2 f1(x) = 1 / (1. + exp[- c1 * (x - b1)]) f2(x) = 1 / (1. + exp[- c2 * (x - b2)])
skfuzzy.membership.gbellmf(x, a, b, c)	Генератор нечеткого членства обобщенной функции Белла.	y(x) = 1 / (1 + abs([x - c] / a) ** [2 * b])
skfuzzy.membership.piecemf(x, abc)	Кусочно-линейная функция принадлежности	y = 0, min(x) <= x <= a y = b(x - a)/c(b - a), a <= x <= b y = x/c, b <= x <= c
skfuzzy.membership.sigmf(x, b, c)	The basic sigmoid membership function generator.	y = 1 / (1. + exp[- c * (x - b)])
skfuzzy.membership.trapmf(x, abcd)	Трапециевидный генератор функций принадлежности.	См. Рисунок 3, б.
skfuzzy.membership.trimf(x, abc)	Треугольная функция принадлежности	См. Рисунок 3, с.

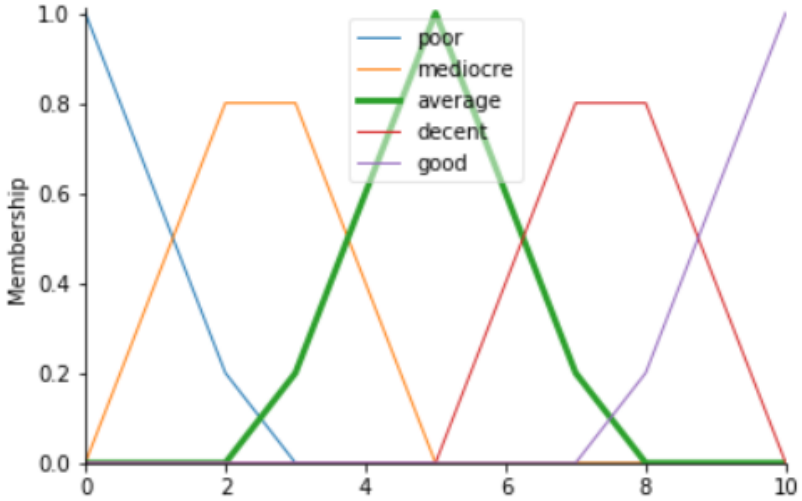
Используются методы генерации, например так:

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

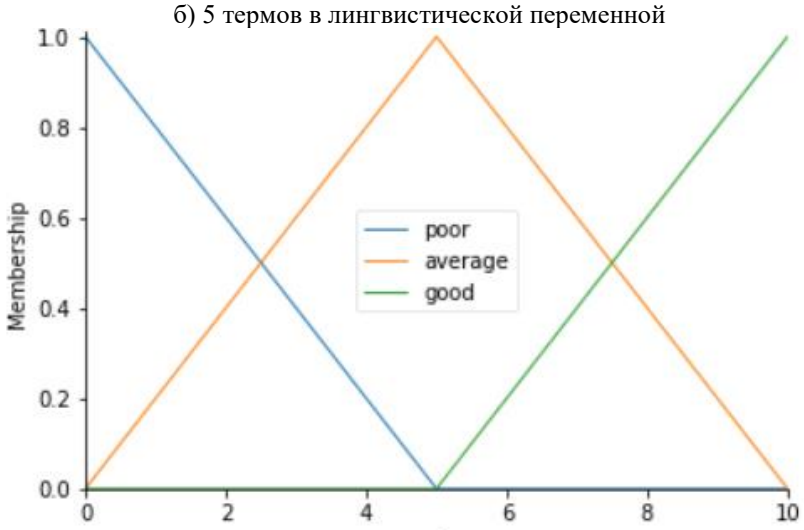
Автоматическое разбиение интервала. При построении функций принадлежности можно использовать автоматическую разбивку интервала на симметричные значения по 3, 5 или 7 термов (Рисунок 3). Имена при этом могут быть заданы автоматически: dismal, poor, mediocre, average, decent, good, excellent при разбиении на 7, - или пользователем. Для задания имён необходимо их определить в виде массива и подать как параметр automf, причем количество элементов массива должно быть также 3, 5, или 7 (например .automf(names= ['nb', 'ns', 'ze', 'ps', 'pb'])) .



а) 7 термов в лингвистической переменной



б) 5 термов в лингвистической переменной



с) 3 терма в лингвистической переменной

Рисунок 3 – Графики функций принадлежности при автоматической разбивке с помощью функции automf

1.5.3 Визуализация лингвистических переменных

Чтобы вывести график функций лингвистической переменной (Рисунок 3,) необходимо вызвать метод view(), для объектов классов `Antecedent или Consequent` (Имя_объекта.view()). Причём, если необходимо выделить конкретный терм, то необходимо указать имя терма (Имя_объекта['имя терма'].view()).

1.6 Определение нечёткой базы знаний (нечёткого контроллера)

1.6.1 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	Обозначение	Пример

оператор		
Или	|	`.Rule(in_p1['poor']|in_p2['poor'], out_p['low'])`

И	&	`.Rule(antecedent=((in_p1['nb'] & in_p2['nb'])`
		`|`
		`(in_p1['ns'] & in_p2['nb'])`
		`|`
		`(in_p1['nb'] & in_p2['ns'])),`
		`consequent=out_p['nb'], label='rule nb')`

Не	~	`.Rule(~ in_p['poor'], out_p['high'])`

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

1.6.2 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*).

1.7 Использование нечёткой системы

1.7.1 Нечёткий вывод

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

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

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

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

Визуализация нечёткого вывода

Для визуализации результата можно использовать метод визуализации выходной нечеткой переменной с указанием в переменной sim объекта ControlSystemSimulation метод (пример view(sim=tipping)).
Для визуализации полной поверхности нечеткого вывда можно построить трехмерный график в случае зависимости одной выходной переменной от двух входных. Для этого можно использовать библиотеку Matplotlib.org (см. подробнее https://matplotlib.org). Ниже в примере приведён код, реализующий такую визуализацию.

1.8 Пример реализации нечёткой системы

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

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

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

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

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

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

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

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

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

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

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

# запускаем расчет
```

```
tipping.compute()

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

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

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

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

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

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

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

Варианты заданий

1. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи закупок (соотношения цены, качества, объема закупок и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
2. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузок спортсмена (соотношение нагрузок, физического состояния,

потребляемых калорий и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

3. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (регулировка скорости с учетом передачи, погодных условий, интенсивности потока и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
4. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи управления транспортным средством (управление рулем, газом, тормозом при въезде в гараж), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
5. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования теплоснабжения (соотношение среднесуточной температуры, ветра, размера здания и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
6. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования реверсного движения на волжском мосту (учитывать время, интенсивность потока, день недели и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
7. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора специй для блюда (соотношение количества и остроты специй, рецептуры, предпочтений едока, объема пищи и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
8. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора объема блюд (учитывать калорийность, вкусовые предпочтения, количество едоков и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
9. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подачи электроэнергии в условиях экономии (учет времени суток, типа помещений, количества людей, типа оборудования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
10. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи подбора интенсивности занятий (учитывать начальный уровень подготовки, объем учебного материала, количество человек в группе, необходимый уровень усвоения и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
11. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи расчета потребления бензина (учитывать тип совершаемых маневров, уровень подготовки водителя, состояние автомобиля, тип автомобиля и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
12. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования системы орошения (учитывать время года, количество выпадающих озадков, вид орошаемой культуры и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
13. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи настройки аудиосистемы (мощность колонок, их количество, размер помещения, назначение установки и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
14. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора дозы снотворного (количество препарата, действие препарата, восприимчивость к выбранному препарату, цель и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

15. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи планирования объема производства продукции (с учетом возможной прибыли, необходимых ресурсов, платежеспособности населения, рынка сбыта и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
16. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи регулирования кондиционера (учитывать его мощность, объем помещения, температуру окружающей среды, необходимую температуру в помещении и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
17. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи распределения нагрузки между компьютерами при использовании их в кластерах (учитывать характеристики компьютеров, их количество, количество параллельного кода, характеристики сети и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
18. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора складского помещения (учитывать площадь склада, количество и размеры продукции, удаленность от места производства и точек реализации, свойства продукции и характеристики помещений и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
19. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи выбора комплектующих для компьютера (учитывать цену, потребности пользователя, совместимость, сроки использования и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).
20. Построить нечеткую базу знаний (использовать не менее 3 лингвистических переменных) для задачи определения количества линий в службе поддержки (учитывать количество обслуживаемых клиентов, среднюю частоту обращения в службу одного клиента, среднее время обслуживания одной заявки, квалификацию персонала и т.д.), проверить ее на полноту и произвести нечеткий вывод для конкретных значений (выбрать случайным образом).

Тема 7. Нейронные сети.

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:

16. Фамилию и номер группы учащегося, задание, вариант

17. Схему НС (ее слоёв)
18. Описание входных данные
19. Описание алгоритма обучения с учетом варианта (функции потерь, оптимизатора и т.д.)
20. Графики динамики обучения НС.
21. Результат тестирования НС.
22. Код.

Методические указания по использованию библиотеки PyTorch для построения нейронных сетей

1.9 Пакет torth.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, сверочная 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).

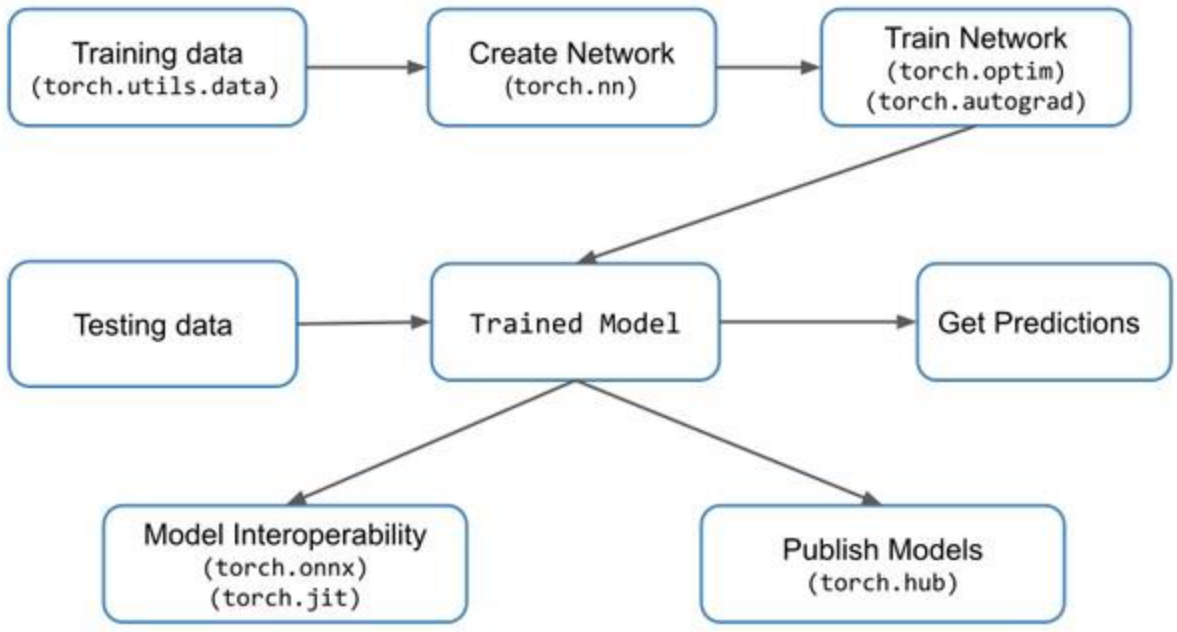


Рисунок 4 – Схема применения модулей PyTorch для обучения НС[2]

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

1.10 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
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

[3] https://neurohive.io/ru/tutorial/cnn-na-pytorch/

	может иметь более одного класса.
LSUN	Используется для крупномасштабной классификации сцен изображений; например, спальня, мост, церковь.
Imagenet-12	Крупномасштабный набор данных визуального распознавания, 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.

1.11 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.

1.12 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.

1.13

1.14 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 **W'x+b**, где W — матрица весов размером *(input, output)* и 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
```

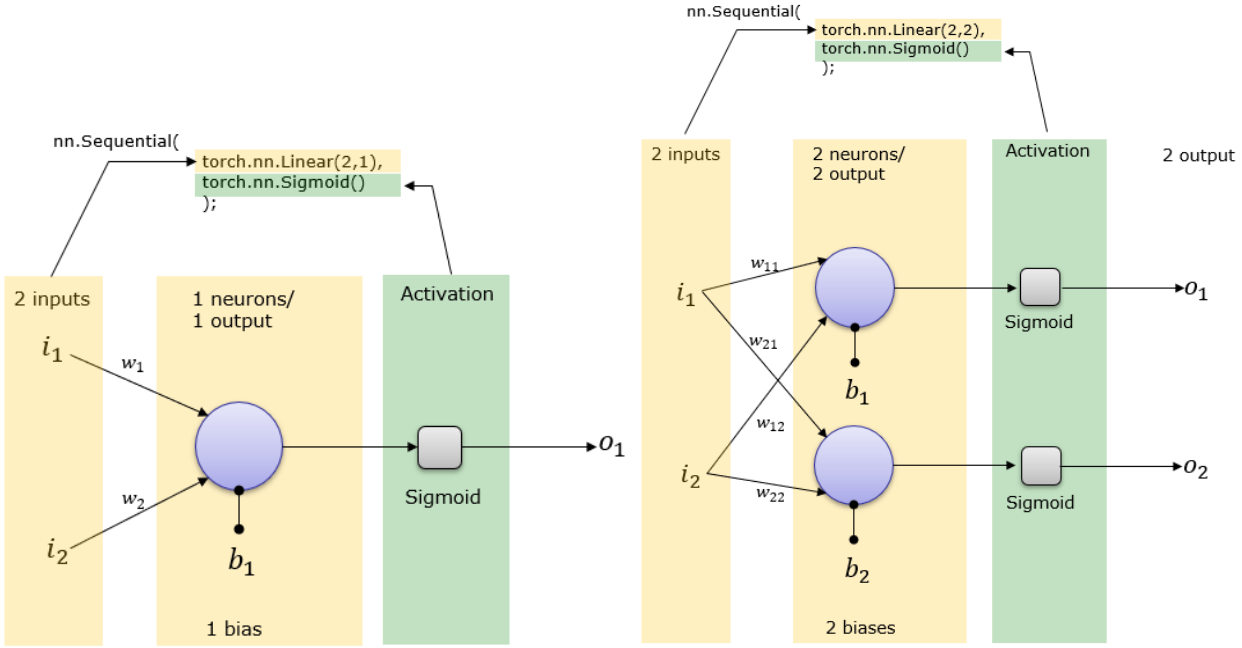
Для построения модели надо создать объект описанного класса:

```
Net=Model()
```

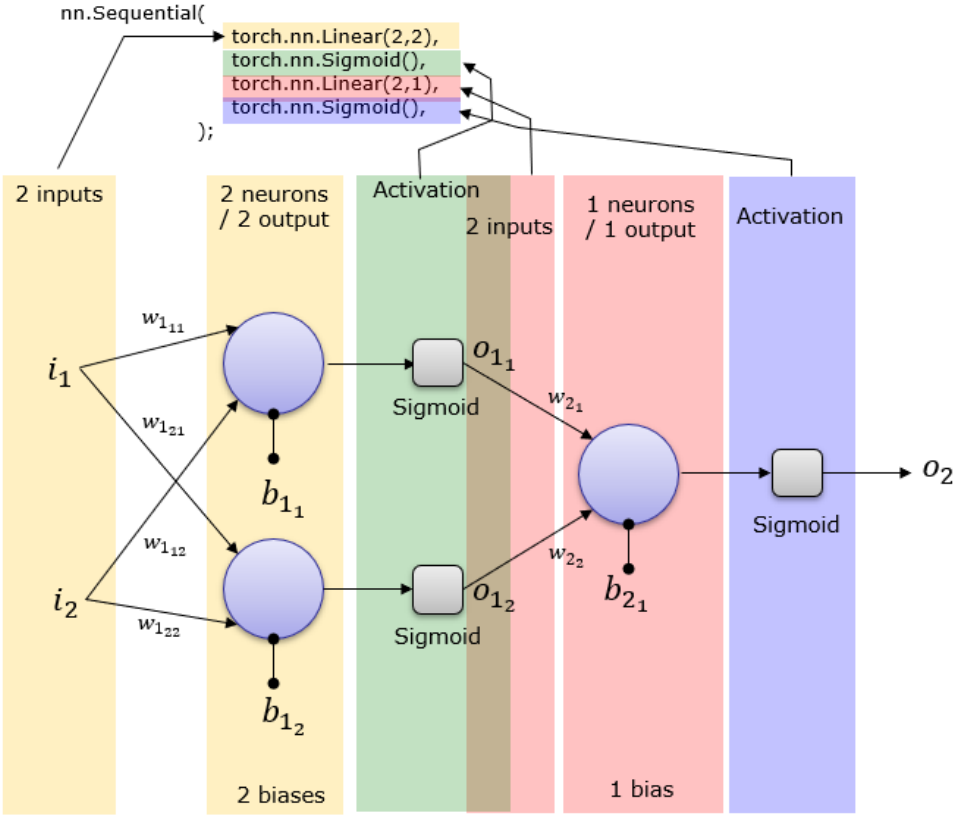
На базе *nn.Sequential*. Контейнер для линейных / последовательных слоёв *Linear*, которые имеют вид **W'x+b**, где W — матрица весов размером *(input, output)* и b — вектор смещения размером *output*. Используя этот контейнер можно в одном операторе определить и вид НС и как будут вычисляться выходные значения.

Пример:

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

Рисунок 5 - Примеры простейшего линейного слоя[4]

Существуют ещё другие возможности по построению НС. В зависимость от вида НС слои (Таблица 14) и функции активации (Таблица 15) могут быть разными., см. подробнее https://pytorch.org/docs/stable/nn.html.

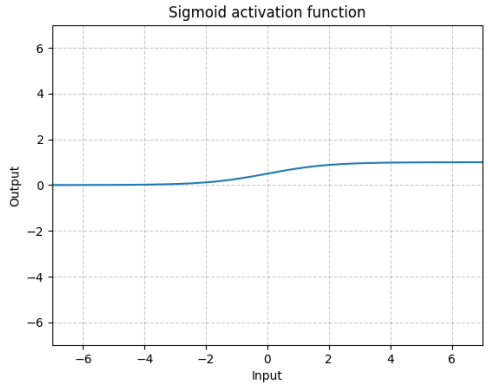
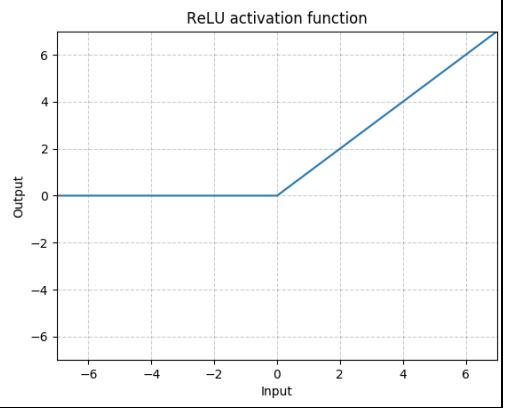
Таблица 14 – Примеры типов слоёв НС

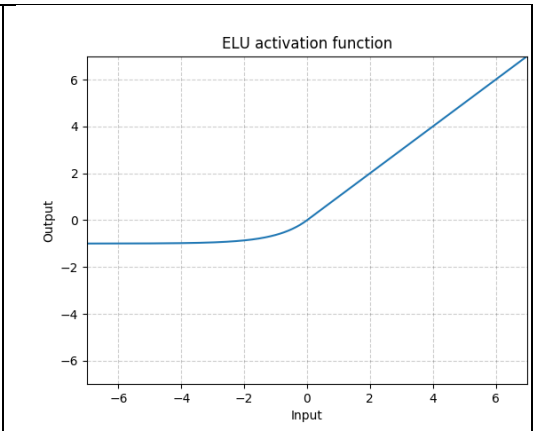
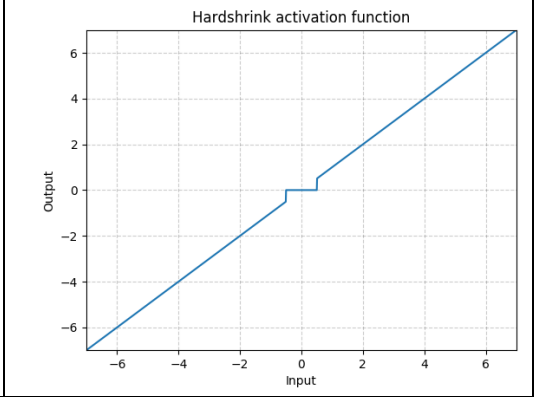
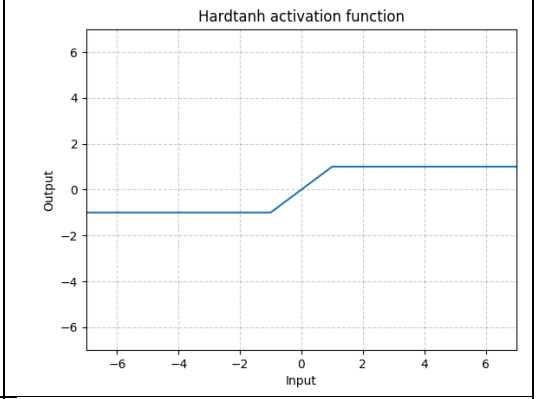
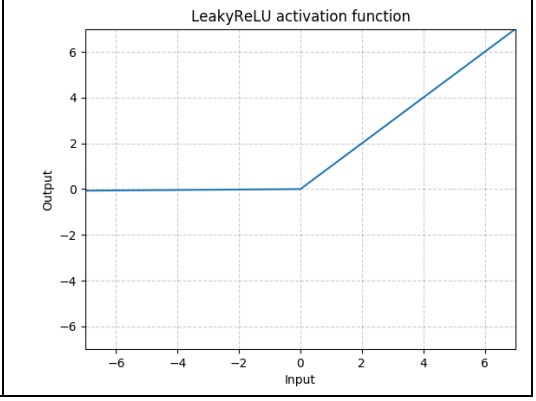
Типы слоев	**Назначение**	**Описание**
Активация (activation)	Общие для НС	Содержит функцию активации, которую применяют ко входам слоя.
Нормализация (Normalization)		Слой обеспечивает применение градиентного спуска не к одной точке выборки, а к небольшой коллекции данных.
Прореживание (dropout), регуляризация		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).
Пуллинг / 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.

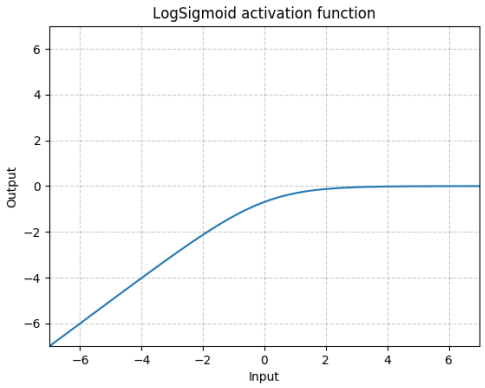
[4] http://www.sharetechnote.com/html/Python_PyTorch_nn_Sequential_01.html

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`	`torch.nn.Sigmoid()` $\text{Sigmoid}(x) = \frac{1}{1 + \exp(-x)}$	
`ReLU`	`torch.nn.ReLU`(*inplace=False*) $\text{ReLU}(x) = \max(0, x)$	

ELU	`torch.nn.ELU`(*alpha=1.0, inplace=False*) $\mathrm{ELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x) - 1))$	
Hardshrink	`torch.nn.Hardshrink`(*lambd=0.5*) $\mathrm{HardShrink}(x) = \begin{cases} x, & \text{if } x > \lambda \\ x, & \text{if } x < -\lambda \\ 0, & \text{otherwise} \end{cases}$	
Hardtanh	`torch.nn.Hardtanh`(*min_val=-1.0, max_val=1.0, inplace=False, min_value=None, max_value=None*) $\mathrm{HardTanh}(x) = \begin{cases} 1 & \text{if } x > 1 \\ -1 & \text{if } x < -1 \\ x & \text{otherwise} \end{cases}$	
LeakyReLU	`torch.nn.LeakyReLU`(*negative_slope=0.01, inplace=False*) $\mathrm{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$	

`LogSigmoid`	`torch.nn.LogSigmoid()` $\text{LogSigmoid}(x) = \log\left(\frac{1}{1+\exp(-x)}\right)$	

1.15 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 `reduction='none'|'mean'|'sum'` $\ell(x, y) = L = \{l_1, \dots, l_N\}^\top$, $\ell(x, y) = \begin{cases} \text{mean}(L), \\ \text{sum}(L), \end{cases}$ Если для поля size_average установлено значение False, потери суммируются для каждой мини-партии. Игнорируется, когда уменьшить является ложным. По умолчанию: True		
`L1Loss`	`torch.nn.L1Loss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = \left\|x_n - y_n\right\|,$
`MSELoss` (средняя квадратичная)	`torch.nn.MSELoss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = (x_n - y_n)^2$

`KLDivLoss` (информационного расхождения Кульбака-Лейблера)	`torch.nn.KLDivLoss`(*size_average=None, reduce=None, reduction='mean'*)	$l_n = y_n \cdot (\log y_n - x_n)$
`BCELoss` (бинарной кросс-энтропии)	`torch.nn.BCELoss`(*weight=None, size_average=None, reduce=None, reduction='mean'*)	$l_n = -w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right],$
`BCEWithLogitsLoss`	`torch.nn.BCEWithLogitsLoss`(*weight=None, size_average=None, reduce=None, reduction='mean', pos_weight=None*)	$l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right],$
`HingeEmbeddingLoss`	`torch.nn.HingeEmbeddingLoss`(*margin=1.0, size_average=None, reduce=None, reduction='mean'*)	$l_n = \begin{cases} x_n, & \text{if } y_n = 1, \\ \max\{0, \Delta - x_n\}, & \text{if } y_n = -1, \end{cases}$

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.
Веса 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 необходмимо на каждой итерации необходимо обнулять градиент, вызывать функцию backward и выполнять шаг оптимизатора.

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

1.16 Использование нейронной сети

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

1.17 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 сигмойда
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)
   # расчитываем 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 использование оптимизатора Adam, с функцией потерь CrossEntropyLoss, обучающая выборка набор MNIST .*

Код:

```
import torch
import torch.nn as nn
```

```
import torchvision.datasets as dsets
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 = dsets.MNIST(
    root='./data',
    train=True,
    transform=transforms.ToTensor(),
    download=True
)
# тестовая выборка
test_dataset = dsets.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_r
ecurrent_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()
        # Прямой проход
        outputs = model(images)
        # Расчитываем функцию потерь
        loss = criterion(outputs, labels)
        # Обратный проход
        loss.backward()
        # Шаг градиентного спуска (изменение параметров НС)
        optimizer.step()
        iter += 1
        if iter % 500 == 0:
            #производим оценки НС
            model.eval()
            correct = 0
            total = 0
            for images, labels in test_loader:
                images = images.view(-1, seq_dim, input_dim)
                outputs = model(images)
                _, predicted = torch.max(outputs.data, 1)
                total += labels.size(0)
                correct += (predicted == labels).sum()
            accuracy = 100 * correct / total
            print('Итерация: {}. Потери: {}. Точность: {}'.format(iter,
loss.item(), accuracy))
```

Варианты заданий

Вариант	Функция потерь	Функция активации	НС	Оптимизатор	Входные данные
1	`L1Loss`	`Sigmoid`	Свёрочная 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	Adam	STL10

			нелинейными слоями (readout)		
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	Свёрочная НС (не менее 2 свёрточных слоёв и одного линейного)	Adam	STL10
7	KLDivLoss	LogSigmoid	Свёрочная 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	Свёрочная 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	Свёрочная 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

*Если параметры варианта не совместимы, обоснуйте и измените.

8. ТЕМАТИКА КУРСОВЫХ, КОНТРОЛЬНЫХ РАБОТ, РЕФЕРАТОВ

Данный вид работы не предусмотрен УП.

9. ПЕРЕЧЕНЬ ВОПРОСОВ К ЭКЗАМЕНУ (ЗАЧЕТУ)

1. Философские вопросы искусственного интеллекта.
2. Системы ИИ. Историческая справка.
3. Основные направления исследований в ИИ.
4. Компьютерная лингвистика: распознавание и синтез речи, машинный перевод.
5. Онтологии. Стандарты Semantic Web.
6. Онтологии: правила и запросы.
7. Знания и данные. Модели представления знаний (логические и продукционные модели).
8. Знания и данные. Модели представления знаний (сетевые и фреймовые модели).

9. Искусственные нейронные сети (основные понятия и определения, виды НС, область применения, отличия архитектуры нейрокомпьютера от архитектуры фон Неймана).
10. Виды функций активаций, инициализация весов, регуляризация нейронных сетей, нормализация в нейронных сетях.
11. Искусственные нейронные сети: алгоритмы обучения (алгоритм обучения по дельта-правилу).
12. Искусственные нейронные сети: алгоритмы обучения (алгоритм обратного распространения ошибки).
13. Свёрточные нейронные сети.
14. Рекуррентные нейронные сети.
15. Теория нечетких множеств (основные понятия и определения, операции над множествами).
16. Теория нечетких множеств (понятие лингвистической переменной, нечеткие высказывания).
17. Генетические алгоритмы (основные понятия и определения, операторы ГА).

10. САМОСТОЯТЕЛЬНАЯ РАБОТА ОБУЧАЮЩИХСЯ

Форма обучения очная

Название разделов и тем	Вид самостоятельной работы	Объем в часах	Форма контроля
Философские вопросы ИИ.	чтение основной и дополнительной литературы, самостоятельное изучение материала по литературным источникам;	4	тестирование
Подходы и направления исследований в ИИ	чтение основной и дополнительной литературы, самостоятельное изучение материала по литературным источникам;	10	тестирование
Представление знаний	самостоятельное выполнение практических заданий репродуктивного типа (ответы на вопросы, тренировочные упражнения, задачи, тесты);	10	Проверка решения задач
Онтологии и Semantic Web	чтение основной и дополнительной литературы, самостоятельное изучение материала по литературным источникам;	12	тестирование
Эволюционное моделирование	самостоятельное выполнение практических заданий репродуктивного типа (ответы на вопросы, тренировочные упражнения, задачи, тесты);	12	Проверка решения задач
Нечеткие вычисления	самостоятельное выполнение практических заданий репродуктивного типа (ответы на вопросы, тренировочные упражнения, задачи, тесты);	12	Проверка решения задач
Нейронные сети	самостоятельное выполнение практических заданий репродуктивного типа (ответы на вопросы, тренировочные упражнения, задачи, тесты);	12	Проверка решения задач

Для самоподготовки использовать (по соответствующим темам):

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

10. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

а) Список рекомендуемой литературы

основная

2) 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
3) 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

4) 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
5) 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
6) 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
7) Исаев, С. В. Интеллектуальные системы : учебное пособие / С. В. Исаев, О. С. Исаева. — Красноярск : Сибирский федеральный университет, 2017. — 120 с. — ISBN 978-5-7638-3781-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/84365.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

9) Липатова С. В. 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» / С. В. Липатова; УлГУ, Фак. математики, информ. и авиац. технологий. - Ульяновск : УлГУ, 2019. - Загл. с экрана; Неопубликованный ресурс. - Электрон. текстовые дан. (1 файл : 2,33 МБ). - Текст : электронный. http://lib.ulsu.ru/ProtectedView/Book/ViewBook/6218

Согласовано:

Главный библиотекарь НБ УлГУ / Полина Н.Ю. /
Должность сотрудника научной библиотеки ФИО подпись

в) Профессиональные базы данных, информационно-справочные системы:

1. Электронно-библиотечные системы:

1.1. **IPRbooks** [Электронный ресурс]: электронно-библиотечная система / группа компаний Ай Пи Эр Медиа . - Электрон. дан. - Саратов, [2019]. - Режим доступа: http://www.iprbookshop.ru.

1.2. **ЮРАЙТ** [Электронный ресурс]: электронно-библиотечная система / ООО Электронное издательство ЮРАЙТ. - Электрон. дан. – Москва, [2019]. - Режим доступа: https://www.biblio-online.ru.

1.3. **Консультант студента** [Электронный ресурс]: электронно-библиотечная система / ООО Политехресурс. - Электрон. дан. – Москва, [2019]. - Режим доступа: http://www.studentlibrary.ru/pages/catalogue.html.

1.4. **Лань** [Электронный ресурс]: электронно-библиотечная система / ООО ЭБС Лань. - Электрон. дан. – С.-Петербург, [2019]. - Режим доступа: https://e.lanbook.com.

1.5. Znanium.com [Электронный ресурс]: электронно-библиотечная система / ООО Знаниум. - Электрон. дан. – Москва, [2019]. - Режим доступа: http://znanium.com.

2. **КонсультантПлюс** [Электронный ресурс]: справочная правовая система. /Компания «Консультант Плюс» - Электрон. дан. - Москва : КонсультантПлюс, [2019].

3. **База данных периодических изданий** [Электронный ресурс] : электронные журналы / ООО ИВИС. - Электрон. дан. - Москва, [2019]. - Режим доступа: https://dlib.eastview.com/browse/udb/12.

4. **Национальная электронная библиотека** [Электронный ресурс]: электронная библиотека. - Электрон. дан. – Москва, [2019]. - Режим доступа: https://нэб.рф.

5. **Электронная библиотека диссертаций РГБ** [Электронный ресурс]: электронная библиотека / ФГБУ РГБ. - Электрон. дан. – Москва, [2019]. - Режим доступа: https://dvs.rsl.ru.

6. Федеральные информационно-образовательные порталы:

6.1. Информационная система Единое окно доступа к образовательным ресурсам. Режим доступа: http://window.edu.ru

6.2. Федеральный портал Российское образование. Режим доступа: http://www.edu.ru

7. Образовательные ресурсы УлГУ:

7.1. Электронная библиотека УлГУ. Режим доступа : http://lib.ulsu.ru/MegaPro/Web

7.2. Образовательный портал УлГУ. Режим доступа : http://edu.ulsu.ru

б) Программное обеспечение

1. Редактор онтологий Protégé (плагины SWRLTab, Pellet): open sourse.
2. Python, Jupiter Notebook, библиотеки pyThorch, dean, Skfuzzy: open sourse.

Согласовано:

Зам. начальника УИТиТ / Клочкова А.В.
Должность сотрудника УИТиТ подпись ФИО

МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ:

Аудитории для проведения лекций (лекционные аудитории 3 корпуса УлГУ), семинарских занятий (лекционные аудитории 3 корпуса УлГУ), для выполнения лабораторных работ и практикумов (дисплейные классы 1 корпуса УлГУ), для проведения текущего контроля и промежуточной аттестации (лекционные аудитории 3 корпуса УлГУ).

Аудитории укомплектованы специализированной мебелью, учебной доской. Аудитории для проведения лекций оборудованы мультимедийным оборудованием для предоставления информации большой аудитории. Помещения для самостоятельной работы оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа к электронной информационно-образовательной среде, электронно-библиотечной системе. Перечень оборудования, используемого в учебном процессе, указывается в соответствии со сведениями о материально-техническом обеспечении и оснащённости образовательного процесса, размещёнными на официальном сайте УлГУ в разделе «Сведения об образовательной организации».

11. СПЕЦИАЛЬНЫЕ УСЛОВИЯ ДЛЯ ОБУЧАЮЩИХСЯ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

В случае необходимости, обучающимся из числа лиц с ограниченными возможностями здоровья (по заявлению обучающегося) могут предлагаться одни из следующих вариантов восприятия информации с учетом их индивидуальных психофизических особенностей:

– для лиц с нарушениями зрения: в форме электронного документа; индивидуальные консультации с привлечением тифлосурдопереводчика; индивидуальные задания и консультации;

– для лиц с нарушениями слуха: в печатной форме; в форме электронного документа; индивидуальные консультации с привлечением сурдопереводчика; индивидуальные задания и консультации;

– для лиц с нарушениями опорно-двигательного аппарата: в печатной форме; в форме электронного документа; индивидуальные задания и консультации.

Разработчик _____ ___доцент_____ С.В. Липатова
подпись должность ФИО