

Мухамедиева Дилноз Тулкуновна

СОВРЕМЕННЫЕ ДОСТИЖЕНИЯ
ПРИМЕНЕНИЯ
ИСКУССТВЕННОГО
ИНТЕЛЛЕКТА ПРИ РЕШЕНИИ
ПРИКЛАДНЫХ ЗАДАЧ

Мухамедиева Дилноз Тулкуновна

СОВРЕМЕННЫЕ ДОСТИЖЕНИЯ
ПРИМЕНЕНИЯ
ИСКУССТВЕННОГО
ИНТЕЛЛЕКТА ПРИ РЕШЕНИИ
ПРИКЛАДНЫХ ЗАДАЧ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ «ТАШКЕНТСКИЙ ИНСТИТУТ
ИНЖЕНЕРОВ ИРРИГАЦИИ И МЕХАНИЗАЦИИ
СЕЛЬСКОГО ХОЗЯЙСТВА»**

**Современные достижения
применения искусственного
интеллекта при решении
прикладных задач**

ТАШКЕНТ – 2023

ИЗДАТЕЛЬСТВО «FAN ZIYOSI»

УДК 330.115

Д.Т.Мухамедиева. «Современные достижения применения искусственного интеллекта при решении прикладных задач». Монография – Т.:Изд.«Fan ziyosi», 2023. 322 с.

В работе рассмотрены актуальные теоретико-методические проблемы применения искусственного интеллекта при решении прикладных задач. Мягкие, или гибридные системы, такие как нечеткие нейронные сети с генетической настройкой параметров, демонстрируют взаимное усиление достоинств и погашение недостатков отдельных методов. Нейронные сети обучаются с помощью универсального алгоритма, то есть трудоемкое извлечение знаний заменяется сбором достаточной по объему обучающей выборки. Изложение ведется в достаточно строгой и в то же время доступной форме. Все основные положения и операции иллюстрируются большим числом примеров. Книга рассчитана на широкий круг читателей, включающий специалистов по прикладной математике, инженеров, а также лиц, интересующихся вопросами обработки или анализа изображений и текстов.

Рекомендовано к печати НТС

Национально- исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Рецензенты:

Маматов Н.С. - д.т.н., профессор Национально- исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Севинов Ж. – д.т.н., профессор Ташкентского государственного технического университета

© Изд.«Fan ziyosi» 2023г.

Содержание

Введение	4
Глава 1. МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА	5
1.1.Нейронные сети	6
1.2. Нечеткие множества	41
1.3. Предварительная обработка и оценка качества изображений на основе аппарата нечетких множеств	78
Глава 2. ПРИМЕНЕНИЕ СОВРЕМЕННЫХ НЕЙРОСЕТЕЙ ПРИ СОЗДАНИИ ИЗОБРАЖЕНИЙ	136
2.1. Как рисовать с помощью нейросетей	136
2.2.Как создать изображения с помощью нейросети	145
2.3. Создание иллюстрации нейросетью	145
Глава3. ПРИМЕНЕНИЕ СОВРЕМЕННЫХ НЕЙРОСЕТЕЙ ПРИ ГЕНЕРИРОВАНИЯ ТЕКСТА	186
3.1.Как нейросеть генерирует текст	186
3.2. Диалог с искусственным интеллектом	212
3.3.Искусственный интеллект составляет программы для решения математических задач	253
3.4. Искусственный интеллект составляет программы для решения задач олимпиады по информатики	298
Заключение	316
Использованная литература	317

ВВЕДЕНИЕ

Искусственный интеллект (ИИ) - это область информационных технологий, которая занимается разработкой программных и аппаратных систем, которые способны имитировать интеллектуальные способности человека. Он обычно включает в себя компьютерные алгоритмы и модели, которые способны обрабатывать данные, анализировать информацию и принимать решения.

Существует множество методов и технологий, которые используются в ИИ. Одним из основных методов является машинное обучение, которое позволяет компьютерным системам "обучаться" на основе большого количества данных и создавать модели, которые могут классифицировать данные, определять шаблоны и прогнозировать будущие события. Другими методами могут быть эволюционные алгоритмы, генетические алгоритмы, искусственные нейронные сети и другие.

Применение ИИ может быть очень разнообразным. Он может использоваться для обработки естественного языка, обнаружения образов, определения эмоций, рекомендации продуктов и услуг, определения рисков и многое другое. ИИ также используется в автономных системах, таких как роботы и беспилотные автомобили, для принятия решений в режиме реального времени.

Одной из главных проблем ИИ является его зависимость от качества данных. Если данные, используемые для обучения системы, недостаточно точны или представляют только небольшую часть возможных сценариев, то ИИ может принимать неверные решения или давать неправильные рекомендации. Кроме того, ИИ может создавать некоторые этические вопросы, например, в области приватности и безопасности.

Несмотря на это, ИИ представляет собой очень важный инструмент для решения сложных задач во многих отраслях, и его значимость только увеличивается в настоящее время.

Глава 1. МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Искусственный интеллект может быть классифицирован на несколько типов, включая слабый ИИ и сильный ИИ. Слабый ИИ ограничен в своих способностях и может выполнять только ограниченный набор задач, для которых он был запрограммирован. Сильный ИИ, с другой стороны, способен выполнять задачи, которые требуют интеллектуальных способностей, эквивалентных или превосходящих те, которые проявляет человек.

Искусственный интеллект также может быть разделен на категории в зависимости от того, какие типы данных он обрабатывает. Некоторые из этих категорий включают в себя компьютерное зрение, обработку естественного языка, машинное обучение и робототехнику.

Компьютерное зрение - это область ИИ, которая позволяет компьютерным системам обрабатывать изображения и видео. Он может быть использован для распознавания лиц, определения объектов на изображениях, и даже для диагностики медицинских изображений.

Обработка естественного языка (Natural Language Processing, NLP) - это область ИИ, которая позволяет компьютерным системам обрабатывать и анализировать естественный язык, используемый людьми. Это может включать в себя распознавание речи, перевод текста с одного языка на другой, анализ тональности и эмоционального содержания текста и многое другое.

Машинное обучение (Machine Learning, ML) - это подход, который позволяет компьютерным системам учиться на основе опыта и использовать этот опыт для решения задач. Он может быть использован для создания предиктивных моделей, кластеризации данных и даже для обучения роботов выполнять определенные задачи.

Робототехника - это область ИИ, которая связана с созданием роботов и автономных систем. Роботы, оснащенные ИИ, могут быть использованы для выполнения множества задач, включая автоматическое управление производственными процессами, исследования местности в межпланетном космосе, и даже для помощи людям в повседневной жизни.

1.1.Нейронные сети

Исследования Мак-Каллока и Питтса в 1943 году легли в основу развития искусственных нейронных сетей. Они предложили математическую модель, описывающую работу биологических нейронов, и показали, что такая модель может быть использована для решения различных задач, таких как логические операции [1-4].

В последующие десятилетия исследования в области нейронных сетей продолжались, и были разработаны различные архитектуры и алгоритмы обучения. Сегодня нейронные сети применяются в различных областях, таких как компьютерное зрение, распознавание речи, машинный перевод, игры и многие другие [1].

Несмотря на то, что нейронные сети показывают впечатляющие результаты в решении задач, связанных с обработкой информации, их возможности все еще ограничены и далеки от полной имитации человеческого интеллекта. Тем не менее, развитие нейронных сетей продолжается, и с каждым годом они становятся все более мощными и универсальными инструментами для решения сложных задач.

Модель Мак-Каллока и Питтса стала отправной точкой для развития двух основных подходов в исследовании нейронных сетей [2].

Первый подход, называемый нейробиологическим, был ориентирован на изучение биологических процессов, лежащих в основе работы нервной системы. Этот подход ставил своей целью создание более точных моделей головного мозга, понимание его функционирования и взаимодействия нейронов между собой.

Второй подход, называемый нейрокомпьютерным, был ориентирован на применение нейронных сетей в качестве метода искусственного интеллекта для решения различных прикладных задач. Этот подход ставил своей целью создание более эффективных и универсальных алгоритмов обучения нейронных сетей, а также их применение в таких областях, как распознавание образов, классификация, анализ данных и другие.

Оба подхода взаимодействуют и взаимообогащают друг друга, благодаря чему нейронные сети продолжают развиваться и находить новые области применения.

Теория Хебба, также известная как правило Хебба, утверждает, что если нейрон А регулярно активируется перед нейроном В и при этом взаимодействие между ними является функционально эффективным, то с течением времени сила связи между ними увеличится. Это означает, что если два нейрона активируются одновременно, то синаптическая связь между ними усилится, что повысит вероятность того, что они будут активироваться вместе в будущем. Этот процесс называется долговременной потенциацией (ДВП) и является основой для запоминания новой информации и обучения [3].

Теория Хебба имеет важные последствия для понимания того, как мозг функционирует и как происходит обучение. Она подразумевает, что мозг обучается путем укрепления связей между нейронами, что позволяет ему запоминать новые знания и умения. Эта теория также объясняет, почему часто повторение задания помогает его запоминанию, так как это укрепляет связи между нейронами.

Теория Хебба была одним из первых научных подходов к пониманию процессов обучения в мозге и оказала значительное влияние на развитие нейронауки и искусственного интеллекта. Ее принципы используются в моделях машинного обучения и нейронных сетях для создания алгоритмов обучения с подкреплением.

В 1954 году Фрэнк Розенблатт, нейрофизиолог и инженер, разработал первую нейронную сеть, которую он назвал "персептрон". Он использовал принципы теории Хебба, чтобы создать модель, которая могла классифицировать изображения на основе определенных параметров [4].

В 1956 году исследователи из Университета Рочестера, включая Алена Ньюэлла и Херберта Саймона, предложили модель нейронной сети, которая называлась "логическим изоморфизмом". Она использовала принципы теории Хебба для создания логических связей между нейронами, что позволяло моделировать процессы принятия решений.

Также в 1956 году Марвин Минский и Сеймур Паперт из Массачусетского технологического института опубликовали статью, в которой они показали, что простые нейронные сети, такие

как персептрон, не могут решать некоторые задачи классификации. Эта работа оказала значительное влияние на исследования в области нейронных сетей и привела к развитию более сложных моделей, таких как многослойные персептроны и сверточные нейронные сети [1-4].

В 1957 году Фрэнк Розенблатт опубликовал статью, в которой он представил модель нейронной сети с двумя слоями, которую он назвал "ADALINE" (Adaptive Linear Neuron). ADALINE была разработана для решения задач классификации и регрессии и использовала обучение с учителем, где модель обучалась на основе предоставленных ей примеров.

В 1958 году Розенблатт представил более сложную модель нейронной сети, которую он назвал "персептрон". Персептрон был также двухслойным, но в отличие от ADALINE, он использовал не только линейные, но и нелинейные функции активации для более сложной обработки информации [1-4].

В 1960 году Розенблатт создал первую работающую модель персептрона, которую он использовал для распознавания образов, написанных на карточках. Эта модель стала одним из первых примеров применения нейронных сетей для решения задач распознавания образов.

Таким образом, можно сказать, что Розенблатт сделал важный вклад в развитие нейронных сетей, создав несколько моделей нейронных сетей и продемонстрировав их работоспособность на практике.

Интерес к исследованию нейронных сетей угас после публикации работы по машинному обучению Минского и Пейперта в 1969 году. Ими были обнаружены основные вычислительные проблемы, возникающие при компьютерной реализации искусственных нейронных сетей. Первая проблема состояла в том, что однослойные нейронные сети не могли совершать "сложение по модулю 2", то есть реализовать функцию "Исключающее ИЛИ". Второй важной проблемой было то, что компьютеры не обладали достаточной вычислительной мощностью, чтобы эффективно обрабатывать огромный объем вычислений, необходимый для больших нейронных сетей [1-4].

Несмотря на возникшие проблемы, в 1980-х годах интерес к искусственным нейронным сетям возродился благодаря развитию

параллельных вычислений и новых методов обучения, таких как обратное распространение ошибки. В 1982 году Джон Хопфилд разработал нейронную сеть, способную обрабатывать информацию и сохранять ее в памяти. В 1986 году Румельхарт и Макклелланд представили работу об использовании параллельно распределенных нейронных сетей для распознавания образов. В дальнейшем нейронные сети получили широкое применение в различных областях, включая распознавание образов, обработку естественного языка, робототехнику, финансовые рынки и др.

Метод обратного распространения ошибки, предложенный Полом Вербосом в 1975 году, позволил решить проблему обучения многослойных нейронных сетей, а также эффективно обрабатывать функции "Исключающее ИЛИ". Это привело к возобновлению интереса к нейронным сетям и ускорению развития этой области исследований [1-4].

Когнитрон, разработанный Фукусимой в 1975 году, был одной из первых многослойных нейронных сетей, способной на обработку изображений. Структура сети и методы настройки весов связей между нейронами в когнитроне варьировались от одной стратегии к другой. Кроме того, каждая из стратегий имела свои преимущества и недостатки.

Несмотря на свою значимость, когнитрон не мог передавать информацию в обоих направлениях. Двусторонняя передача информации между нейронами была достигнута только в сети Хопфилда в 1982 году. Кроме того, в гибридных нейронных сетях была введена специализация узлов для конкретных целей.

Коннективизм - это теория, которая утверждает, что знание в человеческом разуме состоит из связей между нейронами, а не из отдельных элементов. Руммельхарт и Мак-Клелланд использовали коннективизм для создания нейронной сети с распределённым хранением информации, которая могла обучаться распознаванию образов. Эта работа была очень важной в развитии нейронных сетей и считается одной из основных работ в области машинного обучения [1-4].

Вопрос о том, каким образом происходит обучение нейронных сетей в головном мозге, до сих пор остается открытым и вызывает много дискуссий в научном сообществе. Метод обратного распространения ошибки, несмотря на его эффективность при

обучении искусственных нейронных сетей, не дает явного ответа на этот вопрос.

Однако, существуют алгоритмы глубокого обучения, которые позволяют обучать нейронные сети с многими слоями без явного обратного распространения ошибки. Эти алгоритмы, такие как автокодировщики и генеративные модели, позволяют изучать скрытые представления данных, не требуя явного обучающего сигнала. Эти методы нашли применение во многих областях, таких как распознавание образов, обработка естественного языка, генерация изображений и других видов данных.

Таким образом, несмотря на то, что механизмы обучения нейронных сетей в головном мозге все еще не до конца понятны, существующие алгоритмы глубокого обучения предоставляют мощный инструмент для обработки и анализа данных в различных областях.

Искусственные нейронные сети в настоящее время широко применяются в различных областях, таких как компьютерное зрение, обработка естественного языка, робототехника, игры и т.д. И в большинстве случаев они используются в сочетании с другими методами машинного обучения, такими как глубокое обучение, алгоритмы кластеризации, классификации и регрессии, а также с методами оптимизации и поиска.

Также, нейронные сети и другие методы искусственного интеллекта могут использоваться в совместной работе с экспертными системами, базами знаний и другими методами, которые используются для решения задач в разных областях. Это позволяет создавать комплексные системы, которые могут решать задачи с высокой точностью и эффективностью, что делает их важным инструментом для решения сложных проблем в нашем мире.

Сравнение искусственной нейронной сети с человеческим мозгом

Вначале разберем, что из себя представляет наш мозг. В главном органе человеческого тела примерно 86 миллиардов нейронов — клеток, соединенных между собой отростками. Все вместе они представляют огромную сеть, которую называют нейронной. Клетки взаимосвязаны и получают друг от друга

информативные сигналы. Данные обрабатываются и далее по цепочке отправляются другим клеткам.

Искусственная нейронная сеть, в свою очередь, представляет собой математическую модель, которая имитирует работу нейронов и их связей в человеческом мозге. Нейронные сети состоят из множества узлов, которые называются искусственными нейронами, и связей между ними. Каждый нейрон получает входные данные, обрабатывает их и передает результат дальше по сети.

Однако, несмотря на то, что искусственные нейронные сети моделируют работу нейронов, они далеки от полной имитации человеческого мозга. Например, человеческий мозг способен обрабатывать огромные объемы информации одновременно, в то время как искусственная нейронная сеть имеет ограниченную мощность и может обрабатывать только ограниченное количество данных за раз.

Кроме того, человеческий мозг обладает способностью к обучению на основе опыта и адаптации к новым условиям, в то время как искусственная нейронная сеть обычно требует предварительного обучения на большом объеме данных.

Несмотря на это, искусственные нейронные сети имеют множество преимуществ перед человеческим мозгом в определенных задачах, таких как обработка больших объемов данных, распознавание образов, принятие решений на основе определенных правил и т.д.

Центральная нервная система имеет клеточное строение. Единица — нервная клетка, нейрон. Он состоит из тела и отростков, которые соединяют его с внешним миром (рис. 1.1). Отростки, по которым нейрон получает возбуждение, называются дендритами. Отросток, по которому нейрон передает возбуждение, называется аксоном, причем аксон у каждого нейрона один. Дендриты и аксон имеют довольно сложную ветвистую структуру. Место соединения аксона нейрона — источника возбуждения с дендритом называется синапсом. Основная функция нейрона состоит в передаче возбуждения из дендритов в аксон [1-4].

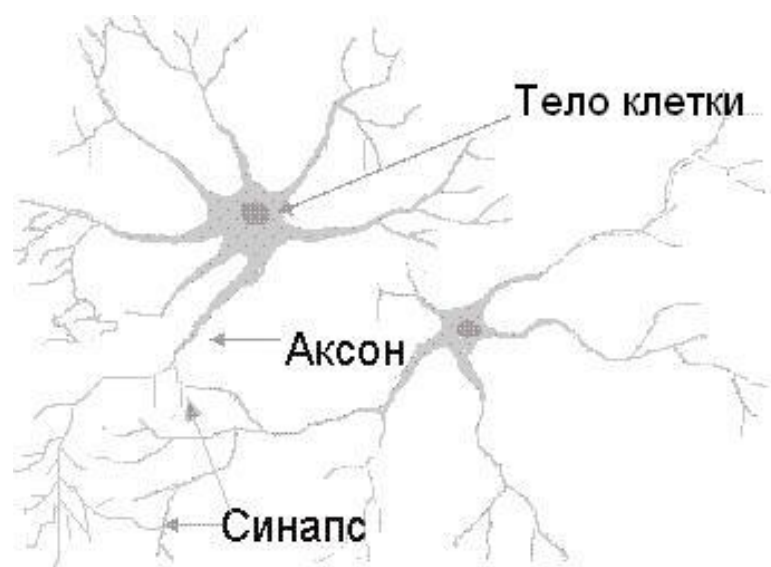


Рис. 1.1 — Модель биологического нейрона

Искусственные нейронные сети имеют схожую структуру с человеческим мозгом. Они состоят из нейронов, которые соединены между собой отростками и образуют сеть. Но в отличие от нейронов в мозге, в искусственных нейронных сетях каждый нейрон имеет несколько входов и один выход. Нейрон получает сигналы от других нейронов через эти входы, и если сумма входных сигналов превышает некоторый порог, нейрон выдает сигнал на выход. Связи между нейронами в искусственных нейронных сетях задаются весами, которые определяют, насколько сильно входной сигнал влияет на нейрон. Эти веса обучаются в процессе обучения сети на некоторых данных.

Хотя искусственные нейронные сети имеют схожую структуру с человеческим мозгом, они далеки от того, чтобы быть такими же эффективными. Мозг является намного более сложной и адаптивной системой, которая способна обучаться на множестве различных задач и быстро приспосабливаться к новым ситуациям. Кроме того, мозг использует не только нейроны, но и другие клетки, такие как глиальные клетки, которые выполняют множество функций, в том числе поддержание нейронов и обеспечение связи между ними.

Нейрон имеет следующие основные свойства:

1. Нейроны являются активными динамическими системами, которые принимают участие в обмене веществ и рассеивают энергию. Они могут изменять свою внутреннюю конфигурацию со временем, реагировать на входные сигналы и формировать

выходные воздействия. Кроме того, нейроны могут обрабатывать информацию, изменять свою чувствительность и подстраиваться к новым условиям, что делает их важными элементами нервной системы. Они также могут обмениваться сигналами с другими нейронами и взаимодействовать в сетях, обеспечивая высокую вычислительную мощность и способность к адаптации.

2. Нейрон обычно имеет множество синапсов - связей с другими нейронами или сенсорными клетками, через которые он передает и получает информацию. Число синапсов на нейрон может варьироваться от нескольких десятков до тысяч, и общее число синапсов в мозге оценивается в десятки триллионов. Каждый синапс состоит из пространства между нейронами, которое называется синаптической щелью, и связанных с ней нейротрансмиттеров - химических веществ, которые используются для передачи информации между нейронами [1-4].

Что такое нейрон?

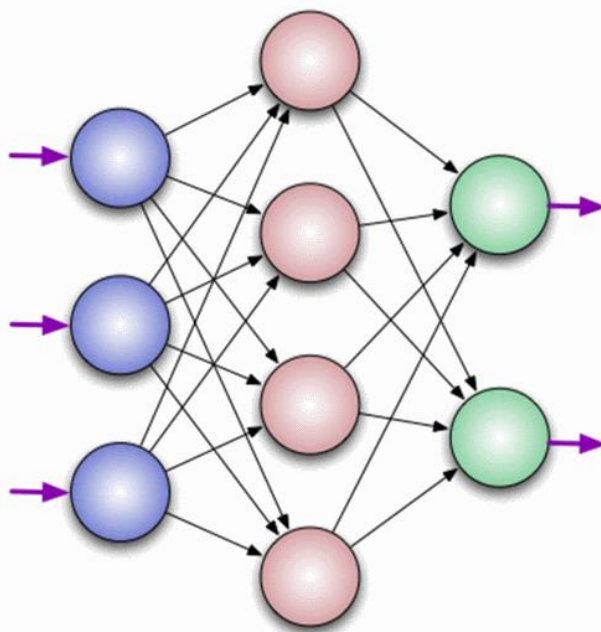


Рис.1.2. Нейрон

Нейрон - это как компьютерный процессор, который принимает информацию, обрабатывает ее и передает дальше. Есть три типа нейронов: входные, скрытые и выходные. Они работают в слоях, где каждый нейрон получает информацию от предыдущего слоя и передает ее дальше. У каждого нейрона есть входные и выходные данные, которые определяются суммой информации от

предыдущего слоя и функцией активации. Функция активации определяет, какая информация будет передана дальше [1-4].

Функция активации - это нелинейная функция, которая добавляет нелинейность в нейронную сеть, что позволяет ей решать более сложные задачи. Кроме того, каждый нейрон имеет свой вес (weight), который определяет, какой вклад в общий результат вносит данный нейрон. В процессе обучения нейронной сети веса оптимизируются, чтобы сеть могла точнее выполнять поставленную задачу.

Для того чтобы нейронная сеть могла обработать данные, важно убедиться, что числа находятся в диапазоне, который она может обработать, а именно $[0,1]$ или $[-1,1]$. Если число выходит за этот диапазон, то его можно нормализовать путем деления 1 на это число. Такая нормализация очень часто используется в нейронных сетях (рис.1.3).

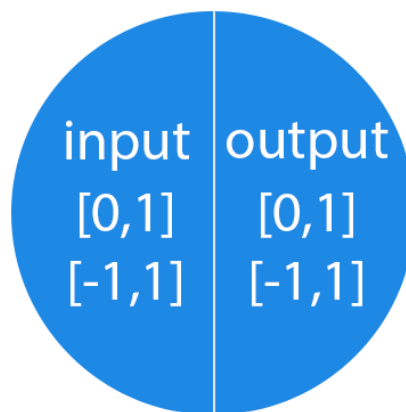


Рис.1.3 - Нормализация в нейронных сетях

Что такое синапс?

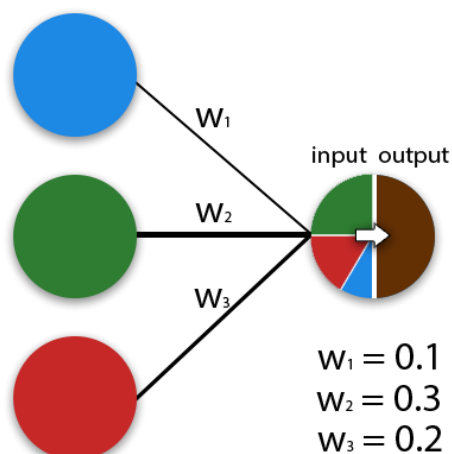


Рис.1.4 - Синапс

Синапсы - это связи между нейронами, которые позволяют передавать информацию от одного нейрона к другому. У каждого синапса есть вес, который определяет важность информации, передаваемой через этот синапс. Например, если вес высокого синапса больше, чем вес других синапсов, то информация, передаваемая через этот синапс, будет более значимой. Эти веса образуют матрицу весов, которая является основой нейронной сети. Эта матрица позволяет обрабатывать входную информацию и производить выходной результат [1-4].

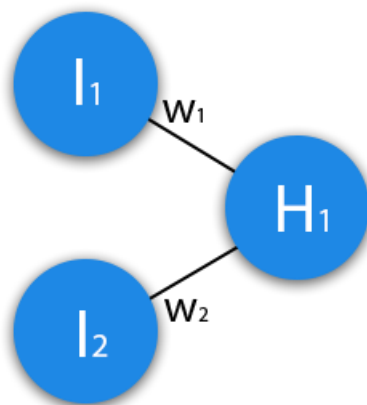
В общем смысле слова, нейронные сети представляют собой математические модели, которые основаны на работе сетей нервных клеток животных. Эти модели могут быть реализованы как в программируемых, так и в аппаратных решениях. Нейрон можно представить как ячейку с множеством входных отверстий и одним выходным. Алгоритм вычисления определяет, как множественные входящие сигналы формируются в выходной сигнал. Каждому входному нейрону подается действительное значение, которое затем передается по межнейронным связям (синапсам). У каждого синапса есть один параметр - вес, который изменяет входную информацию при передаче от одного нейрона к другому. Например, при смешении цветов, разные веса синапсов соответствуют синему, зеленому и красному нейронам. Информация, переданная от нейрона с более высоким весом, будет доминировать в следующем нейроне.

Нейросеть состоит из множества нейронов (или процессоров), которые в отдельности представляют собой простые устройства (значительно проще, чем процессоры персональных компьютеров).

Однако, когда они соединяются в единую систему, они могут выполнять очень сложные задачи.

Как работает нейронная сеть?

В следующем примере изображена часть нейронной сети, где буквами I обозначены входные нейроны, буквой H — скрытый нейрон, а буквой w — веса. Из формулы видно, что входная информация — это сумма всех входных данных, умноженных на соответствующие им веса. Тогда дадим на вход 1 и 0. Пусть $w_1=0.4$ и $w_2 = 0.7$. Входные данные нейрона H1 будут следующими: $1*0.4+0*0.7=0.4$. Теперь когда у нас есть входные данные, мы можем получить выходные данные, подставив входное значение в функцию активации (подробнее о ней далее). Теперь, когда у нас есть выходные данные, мы передаем их дальше. И так, мы повторяем для всех слоев, пока не дойдем до выходного нейрона. Запустив такую сеть в первый раз мы увидим, что ответ далек от правильно, потому что сеть не натренирована. Чтобы улучшить результаты мы будем ее тренировать. Но прежде чем узнать как это делать, давайте введем несколько терминов и свойств нейронной сети [1-4].



$$1) H_{1input} = (I_1 * w_1) + (I_2 * w_2)$$

$$2) H_{1output} = f_{activation}(H_{1input})$$

Рис.1.5 - Часть нейронной сети

Начнем с термина "функция активации". Функция активации является нелинейной функцией, которая определяет выходной сигнал нейрона на основе его входного сигнала. Она позволяет нейронной сети обрабатывать нелинейные зависимости между входными и выходными данными, что делает нейронную сеть более гибкой и способной решать разнообразные задачи.

Другой важный термин - "веса". Веса - это числовые коэффициенты, которые определяют важность входных данных для вычисления выходного значения нейрона. Они являются параметрами нейронной сети, которые настраиваются в процессе обучения сети.

Также стоит упомянуть термин "слои". Нейронные сети состоят из одного или нескольких слоев нейронов. Каждый слой принимает входные данные от предыдущего слоя и генерирует выходные данные для следующего слоя. Существует несколько типов слоев, таких как полносвязные, сверточные, рекуррентные и другие, которые используются в разных типах нейронных сетей.

И, наконец, еще один важный термин - "обучение". Обучение нейронной сети - это процесс настройки ее параметров, таких как веса, чтобы она могла корректно выполнять поставленную перед ней задачу. Обучение может происходить с помощью различных алгоритмов, таких как обратное распространение ошибки (backpropagation), генетические алгоритмы и другие. В процессе обучения нейронная сеть принимает на вход набор обучающих данных и корректирует свои параметры таким образом, чтобы минимизировать ошибку между выходными данными, которые она генерирует, и правильными ответами, заданными в обучающем наборе данных.

Функций активации достаточно много поэтому мы рассмотрим самые основные: Линейная, Сигмоид (Логистическая) и Гиперболический тангенс. Главные их отличия — это диапазон значений.

Функции активации играют очень важную роль в работе нейронных сетей, поскольку они определяют, как будет происходить преобразование входных данных в выходные данные.

Линейная функция активации просто возвращает входное значение, умноженное на некоторый коэффициент (вес). Это означает, что она не изменяет входные данные и не вводит

нелинейность в выходные данные. Эта функция активации обычно используется в задачах регрессии, когда необходимо получить числовой выход.

Сигмоид (Логистическая) функция активации возвращает значение между 0 и 1 и обычно используется в задачах бинарной классификации. Она имеет форму "S"-образной кривой и обеспечивает нелинейность в выходных данных. Однако, у этой функции есть некоторые недостатки, такие как градиентный затухание, который может замедлять процесс обучения в глубоких сетях.

Гиперболический тангенс также возвращает значение между -1 и 1, но в отличие от сигмоидной функции активации, имеет более пологую "S"-образную кривую, что делает ее более подходящей для обучения глубоких нейронных сетей. Она также обеспечивает нелинейность в выходных данных.

Есть и другие функции активации, такие как ReLU (Rectified Linear Unit), которая возвращает 0, если входное значение отрицательно, и возвращает само значение, если оно положительное. Эта функция активации обычно используется в современных нейронных сетях, так как она быстро считается и устраняет проблему затухания градиента.

Тренировочный сеть

Тренировочный сет - это набор данных, который используется для обучения нейронной сети. Он состоит из входных данных (например, изображений или звуковых файлов) и соответствующих им правильных ответов (также называемых метками или метаданными). В процессе тренировки нейронная сеть постепенно корректирует свои веса и параметры, чтобы улучшить свою способность к предсказанию правильных ответов на основе входных данных. Тренировочный сет является ключевым компонентом в обучении нейронных сетей, поскольку он позволяет сети "понимать" структуру входных данных и правильно отвечать на них.

Итерация

Это своеобразный счетчик, который увеличивается каждый раз, когда нейронная сеть проходит один тренировочный сет. Другими словами, это общее количество тренировочных сетов пройденных нейронной сетью.

Итерация - это один полный цикл обучения нейронной сети, когда на вход сети подаются тренировочные данные, происходит вычисление выходных значений, оценка ошибки и корректировка весов сети. Один цикл обучения может включать в себя несколько итераций, когда на каждой итерации нейронная сеть обрабатывает один или несколько тренировочных примеров. В зависимости от алгоритма обучения и размера обучающей выборки, число итераций может быть разным. Например, в стохастическом градиентном спуске каждая итерация может соответствовать одному тренировочному примеру, а в полном градиентном спуске - всей обучающей выборке.

Эпоха

Эпоха - это количество полных проходов набора данных через нейронную сеть в процессе ее обучения. Начиная с нулевой эпохи, с каждым полным проходом набора данных эпоха увеличивается на единицу. Цель увеличения количества эпох - улучшить качество обучения нейронной сети, так как с каждой эпохой сеть получает больше опыта и корректирует свои веса и смещения, приближаясь к оптимальным значениям. Важно помнить, что каждый сет данных должен быть обработан один раз за эпоху, чтобы избежать ошибок в вычислениях.

Ошибка

Ошибка в нейронных сетях представляет собой показатель расхождения между ожидаемым и полученным ответами в процентах. Чтобы эффективно обучать сеть, необходимо следить за тем, чтобы ошибка с каждой эпохой уменьшалась. Для вычисления ошибки можно использовать различные методы, такие как MeanSquaredError (MSE), Root MSE и Arctan. В отличие от функций активации, нет жестких ограничений на выбор метода, и можно выбирать тот, который дает наилучшие результаты. Однако стоит учитывать, что каждый метод вычисляет ошибку по-разному, например, Arctan может давать более высокую ошибку, так как он основан на принципе, что чем больше разница, тем больше ошибка.

MSE обеспечивает баланс в вычислении ошибки, поэтому, чаще всего, используется именно он.

MSE

$$\frac{(i_1-a_1)^2+(i_2-a_2)^2+\dots+(i_n-a_n)^2}{n}$$

Root MSE

$$\sqrt{\frac{(i_1-a_1)^2+(i_2-a_2)^2+\dots+(i_n-a_n)^2}{n}}$$

Arctan

$$\frac{\arctan^2(i_1-a_1)+\dots+\arctan^2(i_n-a_n)}{n}$$

Архитектура базовой нейронной сети

Нейронные сети являются программными моделями, которые имеют структуру, подобную человеческому мозгу. Они состоят из нейронов, которые соединены друг с другом и передают электрические сигналы. В искусственных нейронных сетях также используются искусственные нейроны, которые работают вместе, чтобы решать различные задачи. Эти нейроны являются программными модулями, которые обрабатывают информацию и передают ее другим нейронам для выполнения вычислений. Таким образом, нейронные сети могут использоваться для решения разнообразных задач, используя компьютерные системы для выполнения вычислительных операций.

Нейронная сеть базового уровня обычно состоит из трех слоев взаимодействующих искусственных нейронов:

Входной слой: этот слой получает информацию из внешнего мира и передает ее в нейронную сеть для обработки. Каждый узел

входного слоя обрабатывает данные, классифицирует или анализирует их, затем передает на следующий слой.

Скрытый слой: этот слой получает входные данные от входного слоя или других скрытых слоев. Нейронные сети могут содержать несколько скрытых слоев, каждый из которых анализирует и обрабатывает выходные данные предыдущего слоя и передает результаты на следующий слой.

Выходной слой: этот слой даёт окончательный результат обработки всех данных нейронной сетью. Он может содержать один или несколько узлов. Например, в задаче двоичной классификации (да/нет) выходной слой будет иметь один узел, который даёт результат «1» или «0». В случае множественной классификации выходной слой может содержать несколько узлов.

В чем заключается важность нейронных сетей?

Распознавание общих паттернов.

Нейронные сети могут найти связи и общие паттерны в больших объемах данных. Например, они могут анализировать большие объемы текстовой информации и выявлять наиболее часто встречающиеся слова и фразы, которые могут быть полезны для дальнейшей обработки и анализа.

Распознавание образов и изображений.

Нейронные сети могут обучаться распознавать образы и изображения. Например, они могут классифицировать изображения на основе содержания, например, определять, содержится ли на изображении кошка или собака.

Анализ и предсказание.

Нейронные сети могут использоваться для анализа и предсказания данных. Например, они могут анализировать финансовые данные и предсказывать будущую прибыль компании на основе исторических данных.

В целом, нейронные сети предоставляют мощный инструмент для анализа и обработки данных в различных областях, включая науку, технологии, бизнес и многие другие. Они могут обучаться и адаптироваться к новым данным, что позволяет им постоянно совершенствоваться и улучшать качество своих решений.

Дополнительно, нейронные сети также могут:

Обрабатывать большие объемы данных и выделять наиболее значимые признаки.

Решать задачи, которые традиционно сложны для компьютеров, например, распознавание речи и обработка естественного языка.

Автоматизировать процессы и упрощать сложные задачи, что позволяет повысить эффективность и точность решений в различных сферах, от медицины до финансов.

Создание систем, которые могут учиться самостоятельно и улучшаться с опытом пользователя, является одной из важнейших возможностей нейронных сетей. Например, можно рассмотреть систему автоматического исправления орфографии, которая может адаптироваться к индивидуальным потребностям пользователя, определяя и запоминая часто используемые им слова, даже если они относятся к другому языку.

Нейронные сети могут изучать и моделировать данные, которые сильно меняются со временем. Например, они могут анализировать финансовые транзакции и помечать некоторые из них как мошеннические на основе изменчивых шаблонов. Также они могут обрабатывать сложные данные, такие как белковая структура и ДНК, что может привести к новым открытиям в науке и медицине [1-4].

Для чего используются нейронные сети?

Нейронные сети используются во многих областях для решения различных задач, например:

Для классификации медицинских изображений с целью диагностики;

Для фильтрации социальных сетей и анализа поведенческих данных с целью целевого маркетинга;

Для прогнозирования финансовых инструментов на основе исторических данных;

Для прогнозирования электрической нагрузки и оптимизации энергопотребления;

Для контроля соответствия требованиям и качества;

Для определения химических соединений.

Ниже представлены важнейшие задачи, которые помогают решить нейронные сети.

Машинное зрение

Машинное зрение - это возможность компьютеров анализировать и понимать информацию, содержащуюся в

изображениях и видео. С использованием нейронных сетей компьютеры могут распознавать изображения так же, как это делают люди. Применение машинного зрения распространено в нескольких областях, таких как распознавание дорожных знаков и других участников дорожного движения в беспилотных автомобилях, модерация контента для автоматического удаления небезопасного или неприемлемого контента из архивов изображений и видео, распознавание лиц для идентификации людей и распознавания атрибутов лица, таких как открытые глаза, очки и растительность на лице, а также маркировка изображений для идентификации логотипов бренда, одежды, защитного снаряжения и других деталей изображения [1-4].

Распознавание речи

Нейронные сети способны анализировать человеческую речь независимо от ее моделей, интонации, языка и акцента. Это позволяет виртуальным помощникам, например, Amazon Alexa и программам для автоматической транскрипции, выполнять следующие задачи: помогать операторам колл-центров и автоматически классифицировать звонки, преобразовывать клинические рекомендации в документацию в режиме реального времени, а также создавать точные субтитры к видео и записям совещаний, чтобы сделать контент более доступным [1-4]

Обработка естественного языка (NLP) позволяет компьютерам обрабатывать естественный текст, созданный человеком, и извлекать информацию и смысл из него. С помощью нейронных сетей NLP используется в нескольких областях, включая:

- Автоматизированные виртуальные агенты и чат-боты, которые общаются с людьми на естественном языке
- Автоматическая организация и классификация записанных данных, таких как электронные письма и документы
- Бизнес-аналитика длинных документов, например, автоматическое извлечение ключевых фраз и тем из текстовых данных
- Индексация и анализ настроения, например, для отслеживания публичных мнений о продуктах или услугах в социальных сетях
- Обобщение документов и генерация статей по заданной теме

Многослойные нейронные сети и алгоритмы их обучения

Нейронные сети можно рассматривать как современные вычислительные системы, которые преобразуют информацию по образу процессов, происходящих в мозгу человека. Обработываемая информация имеет численный характер, что позволяет использовать нейронную сеть, например, в качестве модели объекта с совершенно неизвестными характеристиками. Другие типовые приложения нейронных сетей охватывают задачи распознавания, классификации, анализа и сжатия образов.

Свыше 80% всех приложений нейронных сетей относится к так называемым многослойным сетям без обратных связей. В них сигнал пересылается в направлении от входного слоя через скрытые слои (если они имеются) к выходному слою. Сети именно такого типа будут рассматриваться в последующих главах книги в контексте генетических алгоритмов и нечетких систем. Поэтому в настоящей главе мы обсудим в первую очередь базовые элементы многослойных нейронных сетей - персептрон и системы типа Адалайн (с линейным и нелинейным выходом), после чего определим два алгоритма обучения этих сетей: алгоритм обратного распространения ошибки и рекуррентный алгоритм метода наименьших квадратов [1-4].

Нейрон и его модели

Базовый элемент нервной системы - это нервная клетка, называемая нейроном. В нейроне можно выделить тело клетки, называемое сомой, а также исходящие из него два вида отростков: а) по которым в нейрон поступает информация - дендриты и б) по которому нейрон передает информацию - аксон. Каждый нейрон имеет только один выходной отросток, по которому он может передавать импульс нескольким другим нейронам.

Одиночный нейрон принимает возбуждения от огромного количества нейронов (их число может достигать тысячи). Считается, что мозг человека состоит из порядка 10^{11} нейронов, которые имеют между собой примерно 10^{15} соединений. Каждый нейрон передает возбуждение другим нейронам через нервные стыки, называемые синапсами, при этом процесс передачи сигналов имеет сложную электрохимическую природу. Синапсы играют роль репитеров информации, в результате функциони-

рования которых возбуждение может усиливаться или ослабляться. Как следствие, к нейрону приходят сигналы, одна часть из которых оказывает возбуждающее, а вторая - тормозящее воздействие. Нейрон суммирует возбуждающие и тормозящие импульсы. Если их алгебраическая сумма превышает некоторое пороговое значение, то сигнал с выхода нейрона пересылается посредством аксона к другим нейронам.

Рассмотрим модель нейрона, связанную с первыми попытками формализовать описание функционирования нервной клетки. Введем следующие обозначения [1-4]:

u_1, \dots, u_N - входные сигналы данного нейрона, приходящие от других нейронов;

w_1, \dots, w_N — синаптические веса;

y — выходной сигнал нейрона;

v - пороговое значение.

Формула, описывающая функционирование нейрона, имеет вид

$$y = \begin{cases} 1 & \text{при } \sum_{i=1}^N w_i u_i \geq v, \\ 0 & \text{при } \sum_{i=1}^N w_i u_i < v. \end{cases} \quad (1.1)$$

Модель (1.1) может быть представлена в виде

$$y = f\left(\sum_{i=0}^N w_i u_i\right), \quad (1.2)$$

где

$$f(x) = \begin{cases} 1 & \text{при } x \geq 0, \\ 0 & \text{при } x < 0 \end{cases} \quad (1.3)$$

а также $w_0 = v$, $u_0 = 1$.

Формула (1.2) описывает модель нейрона, которая была предложена в 1943 г. МакКаллоком и Питтсом. В качестве функции f может приниматься не только единичная функция (1.3), но и другие пороговые функции вида

$$f(x) = \begin{cases} 1 & \text{при } x \geq 0, \\ -1 & \text{при } x < 0 \end{cases} \quad (1.4)$$

или

$$f(x) = \begin{cases} 1 & \text{при } x > 1, \\ -1 & \text{при } x < -1, \\ x & \text{при } |x| \leq 1. \end{cases} \quad (1.5)$$

На начальной фазе моделирования биологических нейронных сетей применялись пороговые функции (1.3), (1.4) и (1.5). В настоящее время чаще всего используется сигмоидальная функция, определяемая выражением

$$f(x) = \frac{1}{1 + e^{-\beta x}} > 0. \quad (1.6)$$

Отметим, что при $\beta \rightarrow \infty$ характеристика (1.6) стремится к пороговой униполярной функции (1.3). В качестве альтернативы применяется функция гиперболического тангенса

$$f(x) = th\left(\frac{\alpha x}{2}\right) = \frac{1 - e^{-\alpha x}}{1 + e^{-\alpha x}} > 0. \quad (1.7)$$

В этом случае характеристика (1.7) стремится к пороговой биполярной функции (1.4) при $\alpha \rightarrow \infty$.

Персептрон

Модель МакКаллока-Питтса стала отправной точкой для построения простейшей однонаправленной нейронной сети, названной персептроном.

В качестве функции f в модели МакКаллока-Питтса (1.2) применялась биполярная функция активации (1.4). Сигнал x на выходе линейной части персептрона задается выражением

$$x = \sum_{i=1}^N w_i u_i - v = \sum_{i=0}^N w_i u_i, \quad (1.8)$$

где $w_0 = v$, $u_0 = -1$.

Задача персептрона заключается в классификации вектора \mathbf{u} в смысле отнесения его к одному из двух классов, обозначаемых символами L1 и L2. Персептрон относит вектор к классу L1, если выходной сигнал u принимает значение 1, и к классу L2, если выходной сигнал u принимает значение -1. После этого персептрон разделяет N -мерное пространство входных векторов на два полупространства, разделяемые $(N-1)$ -мерной гиперплоскостью, задаваемой уравнением

$$\sum_{i=1}^N w_i u_i - v = \sum_{i=0}^N w_i u_i = 0. \quad (1.9)$$

Гиперплоскость (1.9) называется решающей границей (decision boundary). Если $N = 2$, то решающая граница - это прямая линия, задаваемая уравнением

$$w_1 u_1 + w_2 u_2 - v = 0. \quad (1.10)$$

Точка, лежащая над этой прямой, относится к классу L_1 , тогда как точка, лежащая под этой прямой, относится к классу L_2 . Точки, лежащие на границе решения, можно произвольно отнести и к классу L_1 и к классу L_2 .

Для дальнейших рассуждений допустим, что веса w_i , $i=0,1,\dots,N$ в уравнении гиперплоскости (1.9) неизвестны, тогда как на вход персептрона последовательно подаются так называемые обучающие сигналы $u(n)$, $n = 1, 2, \dots$ где $u(n) = [u_1(n), \dots, u_N(n)]^T$.

Неизвестные значения весов будут определяться в процессе обучения персептрона. Такой подход получил название «обучение с учителем» или «обучение под надзором». Роль «учителя» заключается в корректном отнесении сигналов $u(n)$ к классам L_1 или L_2 , несмотря на неизвестность весов уравнения решающей границы (1.9). По завершении процесса обучения персептрон должен корректно классифицировать поступающие на его вход сигналы, в том числе и те, которые отсутствовали в обучающей последовательности $u(n)$, $n = 1, 2, \dots$. Кроме того, примем, что множества векторов $u(n)$, $n = 1, 2, \dots$, для которых выходной персептрона принимает соответственно значения 1 и -1, линейно отделены, т.е. лежат в двух различных полупространствах, разделенных гиперплоскостью (1.9). Другими словами, допускается разделение обучающей последовательности $\{u(n)\}$ на две последовательности $\{u_1(n)\}$ и $\{u_2(n)\}$ так, что $\{u_1(n)\} \in L_1$ и $\{u_2(n)\} \in L_2$.

В n -й момент времени сигнал на выходе линейной части персептрона определяется выражением

$$x(n) = \sum_{i=0}^N w_i(n) u_i(n) = w^T(n) u(n), \quad (1.11)$$

где

$$u(n) = [-1, u_1(n), u_2(n), \dots, u_N(n)]^T, \quad (1.12)$$

$$w(n) = [v(n), w_1(n), w_2(n), \dots, w_N(n)]^T, \quad (1.13)$$

Обучение персептрона заключается в рекуррентной коррекции вектора весов $w(n)$ согласно формулам

$$w(n+1) = \begin{cases} w(n), & \text{если } w^T(n)u(n) \geq 0 \text{ и } u(n) \in L_1, \\ w(n), & \text{если } w^T(n)u(n) < 0 \text{ и } u(n) \in L_2, \end{cases} \quad (1.14)$$

и

$$w(n+1) = \begin{cases} w(n) - \eta u(n), & \text{если } w^T(n)u(n) \geq 0 \text{ и } u(n) \in L_2, \\ w(n) + \eta u(n), & \text{если } w^T(n)u(n) < 0 \text{ и } u(n) \in L_1, \end{cases} \quad (1.15)$$

где параметр η при $0 < \eta < 1$ - шаг коррекции, тогда как начальные значения компонент вектора весов устанавливаются равными нулю, т.е.

$$w(0) = 0. \quad (1.16)$$

Зависимости (1.14) и (1.15) можно представить в более сжатом виде. Для этого определим так называемый эталонный (заданный) сигнал $d(n)$ в форме

$$d(n) = \begin{cases} +1, & \text{если } u(n) \in L_1, \\ -1, & \text{если } u(n) \in L_2. \end{cases} \quad (1.17)$$

Кроме того, отметим, что выходной сигнал персептрона может быть описан выражением

$$y(n) = \text{sgn}(w^T(n)u(n)). \quad (1.18)$$

С учетом введенных обозначений рекурсии (1.14) и (1.15) принимают вид

$$w(n+1) = w(n) + [d(n) - y(n)]u(n). \quad (1.19)$$

Разность $d(n) - y(n)$ можно интерпретировать как погрешность между эталонным (заданным) сигналом $d(n)$ и фактическим выходным сигналом $y(n)$.

С учетом принятого выше условия линейной сепарабельности входных сигналов алгоритм (1.19) сходится, т.е.

$$w(n_0) = w(n_0 + 1) = w(n_0 + 2) = \dots \quad (1.20)$$

По завершении обучения решающая граница персептрона определяется выражением

$$\sum_{i=0}^N w_i(n_0)u_i = 0, \quad (1.21)$$

а персептрон корректно классифицирует как сигналы, которые принадлежат к обучающей выборке $\{u(n)\}$, так и не входящие в это множество, но выполняющие условие линейной сепарабельности. Напомним, что условию линейной сепарабельности не отвечает логическая функция XOR, заданная таблицей 1.1.

Из рис. 1.6 следует, что не существует прямой, которая отделила бы точки со значениями функции XOR, равными -1 , от точек со значениями, равными 1 . В этом случае роль примерной границы играет эллипс, и поэтому алгоритм (1.18) не был бы сходящимся. Проблему XOR можно разрешить с помощью двухслойного персептрона.

Таблица 1.1. Логическая функция XOR

u1	u2	d=XOR(u1,u2)
+1	+1	-1
+1	-1	+1
-1	+1	+1
-1	-1	-1

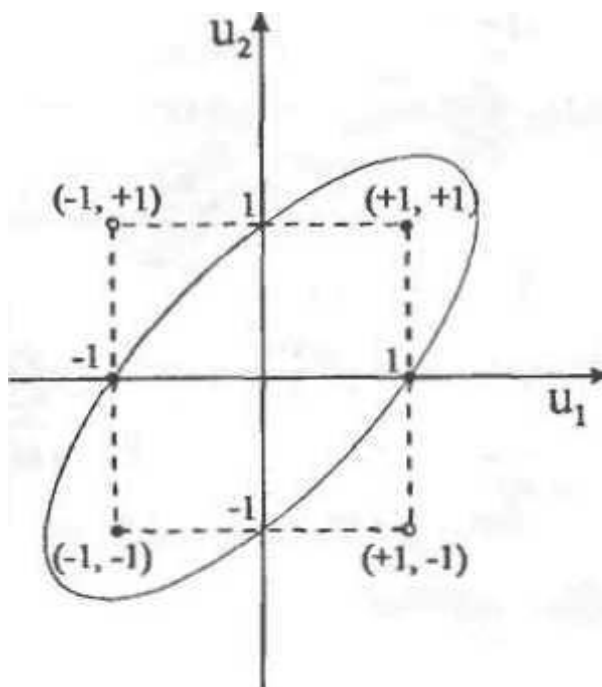


Рис. 1.6. Иллюстрация проблемы XOR

Линейный взвешенный сумматор

На рис. 1.7 представлена структура линейного взвешенного сумматора (linear combiner). Его выход образуется сигналом $y(n)$, который представляет собой линейную комбинацию всех входов $u_1(n), u_2(n), \dots, u_N(n)$, $n = 1, 2, \dots$. Введем обозначение [1-4]

$$u(n) = [u_1(n), u_2(n), \dots, u_N(n)]^T. \quad (1.22)$$

Конкретные компоненты вектора $u(n)$ умножаются на компоненты вектора весов

$$w = [w_1, w_2, \dots, w_N]^T. \quad (1.23)$$

В результате выходной сигнал линейного взвешенного сумматора описывается формулой

$$y(n) = \sum_{k=1}^M w_k u_k(n) = w^T u(n). \quad (1.24)$$

Выход линейного взвешенного сумматора $y(n)$ будет использоваться в качестве реализации некоторого сигнала $d(n)$, называемого эталонным или заданным сигналом. В результате сравнения реализации $y(n)$ с сигналом $d(n)$ получаем погрешность реализации

$$\varepsilon(n) = d(n) - y(n). \quad (1.25)$$

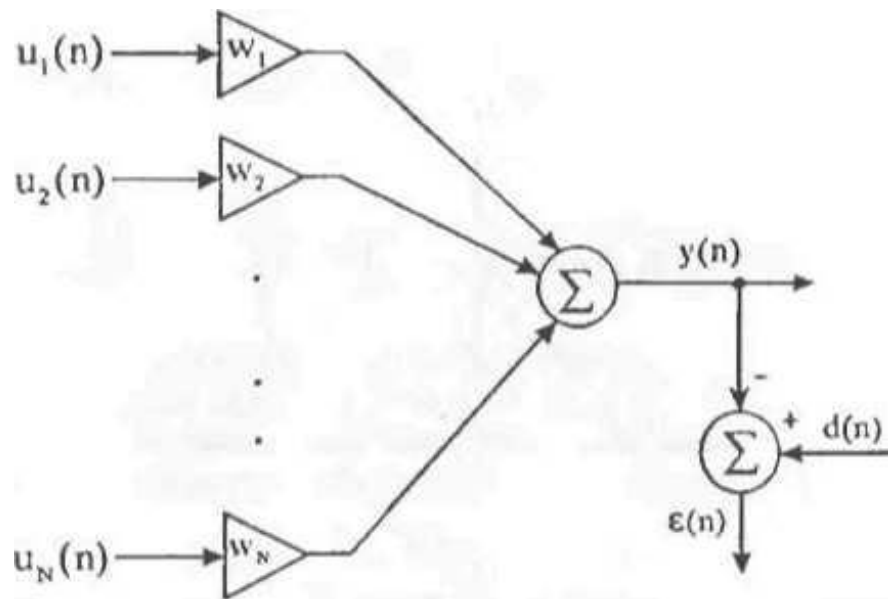


Рис. 1.7. Линейный взвешенный сумматор.

Веса линейного взвешенного сумматора w_1, w_2, \dots, w_N будут подбираться так, чтобы минимизировать меру погрешности

$$Q(w) = E[\varepsilon(n)^2] = E\left[\left(d(n) - \sum_{k=1}^N w_k u_k(n)\right)^2\right] = E[(d(n) - w^T u(n))^2]. \quad (1.26)$$

Предположим, что входной сигнал $u(n)$ и эталонный сигнал $d(n)$ - это реализации дискретных стохастических процессов $\{u(n)\}$ и $\{d(n)\}$, совместно стационарных в широком смысле, т.е.

$\{u(n)\}$ - стационарный в широком смысле стохастический процесс;

$\{d(n)\}$ - стационарный в широком смысле стохастический процесс;

функция взаимной корреляции процессов $\{u(n)\}$ и $\{d(n - k)\}$ зависит только от значения k .

Мера погрешности (1.26) называется среднеквадратичной погрешностью реализации. Обозначим $[w'_1, \dots, w'_N]^T = w'$ вектор весов, минимизирующий погрешность (1.26). Представленная на рис. 1.7 система, веса которой принимают значения w'_1, \dots, w'_N называется пространственным фильтром Винера (spatial filter). Процесс фильтрации заключается в умножении входов $u_1(n), \dots, u_N(n)$ на соответствующее им множество весов w'_1, \dots, w'_N с последующим суммированием отдельных произведений для получения реализации $y(n)$ эталонного сигнала $d(n)$. Покажем, что среднеквадратичная погрешность реализации (1.26) - это функция второго порядка вектора весов w .

Поскольку

$$[d(n) - w^T u(n)]^2 = d^2(n) - 2d(n)w^T u(n) + w^T u(n)u^T(n)w \quad (1.27)$$

то формула (1.26) принимает вид

$$Q(w) = E[d^2(n)] - 2w^T E[d(n)u(n)] + w^T E[u(n)u^T(n)]w. \quad (1.28)$$

В последнем слагаемом выражения (2.28) можно выделить матрицу автокорреляции компонентов входного вектора

$$R = E[u(n)u^T(n)] = \begin{bmatrix} E[(u_1^2(n))] & E[u_1(n)u_2(n)] & \dots & E[u_1(n)u_N(n)] \\ E[u_2(n)u_1(n)] & E[(u_2^2(n))] & \dots & E[u_2(n)u_N(n)] \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ E[u_N(n)u_1(n)] & E[u_N(n)u_2(n)] & \dots & E[(u_N^2(n))] \end{bmatrix}, \quad (1.29)$$

а также вектор взаимной корреляции между сигналами $d(n)$ и $u(n)$

$$p = E[d(n)u(n)] = \begin{bmatrix} E[d(n)u_1(n)] \\ E[d(n)u_2(n)] \\ \cdot \\ \cdot \\ \cdot \\ E[d(n)u_N(n)] \end{bmatrix}. \quad (1.30)$$

Кроме того, доказано, что скорость сходимости алгоритма наискорейшего спуска зависит от отношения наименьшего и наибольшего собственных значений матрицы R . Если

$$\frac{\lambda_{\min}}{\lambda_{\max}} \cong 1,$$

то алгоритм наискорейшего спуска сходится быстро. Если же

$$\frac{\lambda_{\min}}{\lambda_{\max}} \cong 0.$$

то алгоритм наискорейшего спуска сходится медленно.

Адаптивный линейный взвешенный сумматор

Применение алгоритма предполагает знание матрицы R и вектора \hat{d} . В случае, когда эти величины неизвестны, следует заменить градиент его приближением. Запишем рекурсивное выражение в виде

$$w(n+1) = w(n) - \frac{1}{2} \eta \frac{\partial E[\varepsilon^2(n)]}{\partial w(n)}.$$

Если в этой формуле заменить градиент его приближенным локальным значением (instantaneous estimate), т.е.

$$\frac{\partial E[\varepsilon^2(n)]}{\partial w(n)} \rightarrow \frac{\partial \varepsilon^2(n)}{\partial w(n)}.$$

то получим рекурсию вида

$$\hat{w}(n+1) = \hat{w}(n) - \frac{1}{2} \eta \frac{\partial \varepsilon^2(n)}{\partial w(n)}.$$

Нейронные сети, в зависимости от их структуры и способа обработки данных, могут быть разделены на различные типы. Одним из самых распространенных и важных типов нейронных сетей являются сверточные нейронные сети (СНС). Они широко

применяются для анализа изображений и видео, а также обработки языковых данных.

Основные характеристики сверточных нейронных сетей включают способность обрабатывать большие объемы данных, использование нейронов в трехмерном пространстве и механизм определения местоположения для работы нелинейных фильтров. Идея создания сверточных нейронных сетей основана на изучении зрительной коры мозга, которая играет ключевую роль в процессе визуализации.

СНС - это важный инструмент в области глубокого обучения, который позволяет автоматически распознавать и анализировать сложные данные, такие как изображения и видео. Этот тип нейронных сетей используется во многих областях, таких как медицинская диагностика, автоматическое распознавание лиц, автоматическое управление транспортными средствами и других.

Сверточные нейронные сети обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. Сверточные нейронные сети объединяют три архитектурных идеи, для обеспечения инвариантности к изменению масштаба, повороту сдвигу и пространственным искажениям:

- локальные рецепторные поля (обеспечивают локальную двумерную связность нейронов);
- общие синаптические коэффициенты (обеспечивают детектирование некоторых черт в любом месте изображения и уменьшают общее число весовых коэффициентов);
- иерархическая организация с пространственными подвыборками.

На данный момент сверточная нейронная сеть и ее модификации считаются лучшими по точности и скорости алгоритмами нахождения объектов. Начиная с 2012 года, нейросети занимают первые места на известном международном конкурсе по распознаванию образов ImageNet.

Именно поэтому использовался сверточная нейронная сеть, основанную на принципах неокогнитрона и дополненную обучением по алгоритму обратного распространения ошибки.

Структура сверточной нейронной сети

СНС состоит из разных видов слоев: сверточные (convolutional) слои, субдискретизирующие (subsampling, подвыборка) слои и слои «обычной» нейронной сети – перцептрона, в соответствии с рисунком 1.8.

Первые два типа слоев (convolutional, subsampling), чередуясь между собой, формируют входной вектор признаков для многослойного перцептрона.

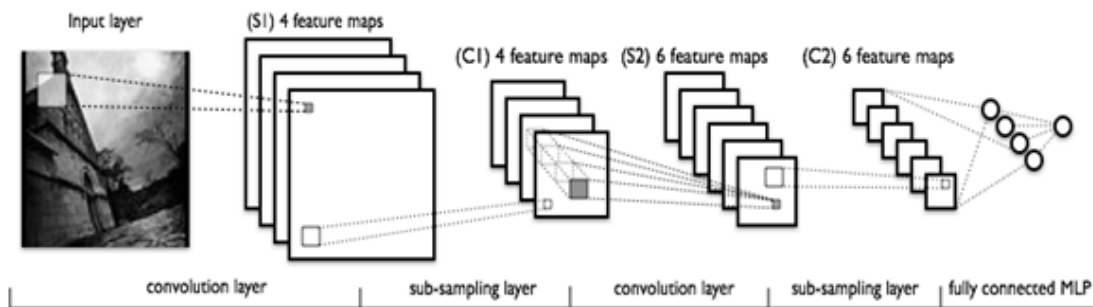


Рис. 1.8 – топология сверточной нейронной сети

Общая топология сети в соответствии с рисунком 1.9.

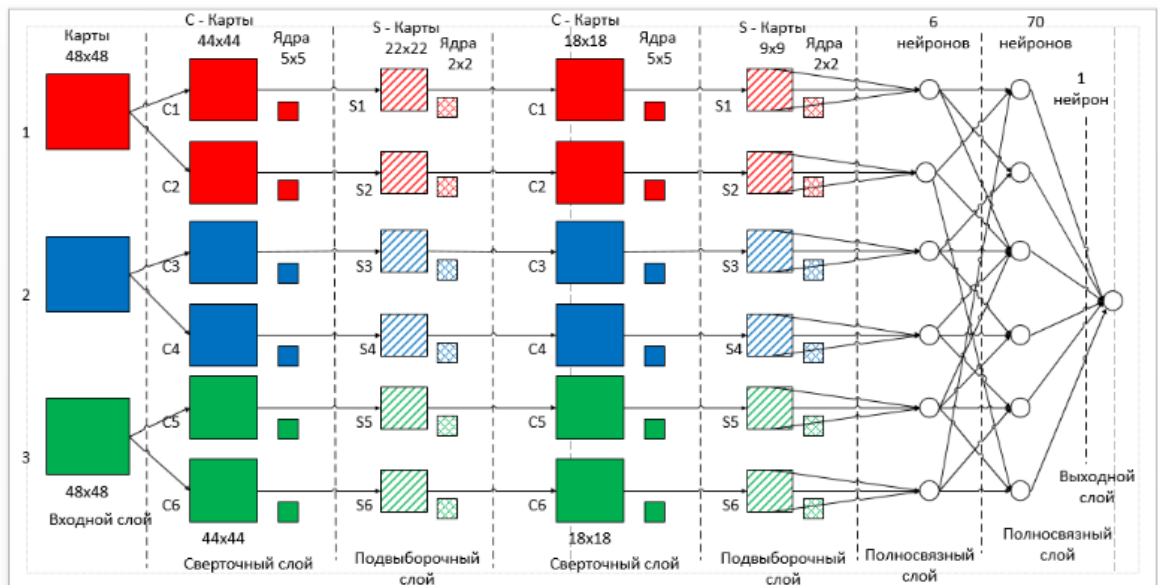


Рисунок 2 — Топология сверточной нейросети

Рис.1.9 — Топология сверточной нейросети

Входной слой

Входные данные представляют из себя цветные изображения типа JPEG, размера 48x48 пикселей. Если размер будет слишком велик, то вычислительная сложность повысится, соответственно ограничения на скорость ответа будут нарушены, определение размера в данной задаче решается методом подбора. Если выбрать размер слишком маленький, то сеть не сможет выявить ключевые признаки лиц. Каждое изображение разбивается на 3 канала: красный, синий, зеленый. Таким образом получается 3 изображения размера 48x48 пикселей.

Входной слой учитывает двумерную топологию изображений и состоит из нескольких карт (матриц), карта может быть одна, в том случае, если изображение представлено в оттенках серого, иначе их 3, где каждая карта соответствует изображению с конкретным каналом (красным, синим и зеленым).

Входные данные каждого конкретного значения пикселя нормализуются в диапазон от 0 до 1, по формуле:

$$f(p, min, max) = \frac{p - min}{max - min},$$

где f – функция нормализации;

p – значение конкретного цвета пикселя от 0 до 255;

min – минимальное значение пикселя – 0;

max – максимальное значение пикселя – 255.

Сверточный слой

Сверточный слой представляет из себя набор карт (другое название – карты признаков, в обиходе это обычные матрицы), у каждой карты есть синаптическое ядро (в разных источниках его называют по-разному: сканирующее ядро или фильтр).

Количество карт определяется требованиями к задаче, если взять большое количество карт, то повысится качество распознавания, но увеличится вычислительная сложность. Исходя из анализа научных статей, в большинстве случаев предлагается брать соотношение один к двум, то есть каждая карта предыдущего

слоя (например, у первого сверточного слоя, предыдущим является входной) связана с двумя картами сверточного слоя, в соответствии с рисунком 1.10. Количество карт – 6.

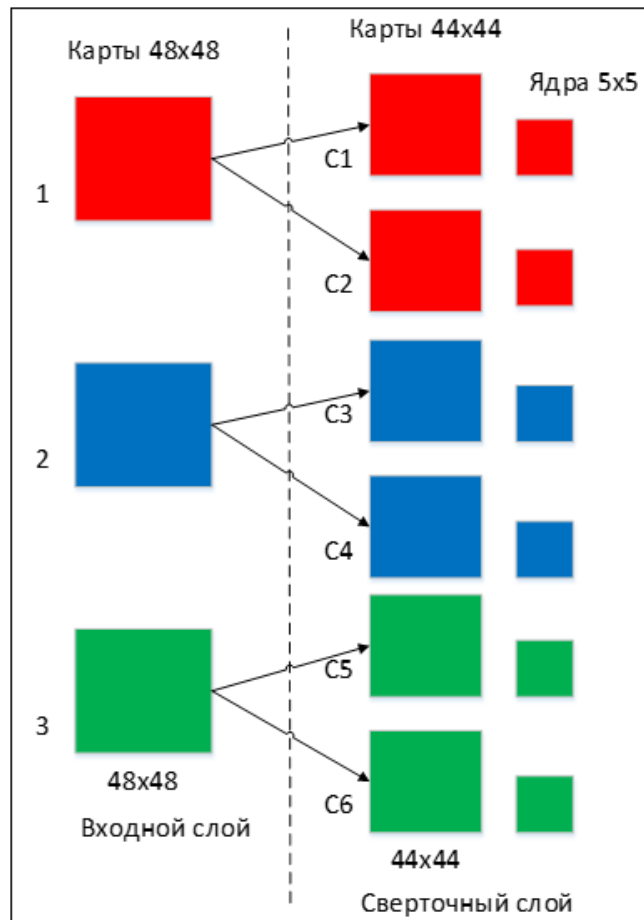


Рис. 1.10 — Организация связей между картами сверточного слоя и предыдущего

Размер у всех карт сверточного слоя – одинаковы и вычисляются по формуле:

$$(w, h) = (mW - kW + 1, mH - kH + 1),$$

где (w, h) – вычисляемый размер сверточной карты;

mW – ширина предыдущей карты;

mH – высота предыдущей карты;

kW – ширина ядра;

kH – высота ядра.

Ядро представляет из себя фильтр или окно, которое скользит по всей области предыдущей карты и находит определенные признаки объектов. Например, если сеть обучали на множестве лиц, то одно из ядер могло бы в процессе обучения выдавать наибольший сигнал в области глаза, рта, брови или носа, другое ядро могло бы выявлять другие признаки. Размер ядра обычно берут в пределах от 3x3 до 7x7. Если размер ядра маленький, то оно не сможет выделить какие-либо признаки, если слишком большое, то увеличивается количество связей между нейронами. Также размер ядра выбирается таким, чтобы размер карт сверточного слоя был четным, это позволяет не терять информацию при уменьшении размерности в подвыборочном слое, описанном ниже.

Ядро представляет собой систему разделяемых весов или синапсов, это одна из главных особенностей сверточной нейросети. В обычной многослойной сети очень много связей между нейронами, то есть синапсов, что весьма замедляет процесс детектирования. В сверточной сети – наоборот, общие веса позволяет сократить число связей и позволить находить один и тот же признак по всей области изображения.

Изначально значения каждой карты сверточного слоя равны 0. Значения весов ядер задаются случайным образом в области от -0.5 до 0.5. Ядро скользит по предыдущей карте и производит операцию свертки, которая часто используется для обработки изображений, формула:

$$(f * g)[m, n] = \sum_{k, l} f[m - k, n - l] * g[k, l],$$

где f – исходная матрица изображения;

g – ядро свертки.

В упрощенном виде этот слой можно описать формулой:

$$x^l = f(x^{l-1} * k^l + b^l),$$

- где x^l – выход слоя l ;
 $f()$ – функция активации;
 b^l – коэффициент сдвига слоя l ;
 $*$ – операция свертки входа x с ядром k .

При этом за счет краевых эффектов размер исходных матриц уменьшается, формула:

$$x_j^l = f\left(\sum_i x_i^{l-1} * k_j^l + b_j^l\right),$$

- где x_j^l – карта признаков j (выход слоя l);
 $f()$ – функция активации;
 b^l – коэффициент сдвига слоя l для карты признаков j ;
 k_j^l – ядро свертки j карты, слоя l ;
 $*$ – операция свертки входа x с ядром k .

Подвыборочный слой

Подвыборочный слой также, как и сверточный имеет карты, но их количество совпадает с предыдущим (сверточным) слоем, их 6. Цель слоя – уменьшение размерности карт предыдущего слоя. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться.

В процессе сканирования ядром подвыборочного слоя (фильтром) карты предыдущего слоя, сканирующее ядро не

пересекается в отличие от сверточного слоя. Обычно, каждая карта имеет ядро размером 2×2 , что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Вся карта признаков разделяется на ячейки 2×2 элемента, из которых выбираются максимальные по значению.

Обычно в подвыборочном слое применяется функция активации RelU. Операция подвыборки (или MaxPooling – выбор максимального) в соответствии с рисунком 1.11.

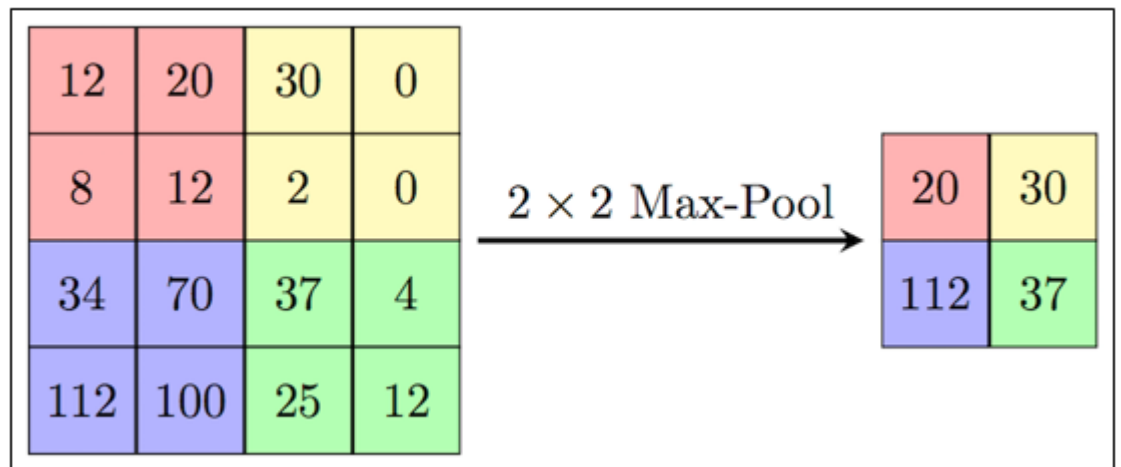


Рис. 1.11 — Формирование новой карты подвыборочного слоя на основе предыдущей карты сверточного слоя. Операция подвыборки (Max Pooling)

Формально слой может быть описан формулой:

$$x^l = f(a^l * \text{subsample}(x^{l-1}) + b^l),$$

где x^l – выход слоя l ;

$f()$ – функция активации;

a^l, b^l – коэффициенты сдвига слоя l ;

$\text{subsample}()$ – операция выборки локальных максимальных значений.

Полносвязный слой

Последний из типов слоев это слой обычного многослойного персептрона. Цель слоя – классификация, моделирует сложную нелинейную функцию, оптимизируя которую, улучшается качество распознавания.

Нейроны каждой карты предыдущего подвыборочного слоя связаны с одним нейроном скрытого слоя. Таким образом число нейронов скрытого слоя равно числу карт подвыборочного слоя, но связи могут быть не обязательно такими, например, только часть нейронов какой-либо из карт подвыборочного слоя быть связана с первым нейроном скрытого слоя, а оставшаяся часть со вторым, либо все нейроны первой карты связаны с нейронами 1 и 2 скрытого слоя. Вычисление значений нейрона можно описать формулой:

$$x_j^l = f \left(\sum_i x_i^{l-1} * w_{i,j}^{l-1} + b_j^{l-1} \right),$$

где x_j^l – карта признаков j (выход слоя l);

$f()$ – функция активации;

b^l – коэффициент сдвига слоя l ;

$w_{i,j}^l$ – матрица весовых коэффициентов слоя l .

Выходной слой

Выходной слой связан со всеми нейронами предыдущего слоя. Количество нейронов соответствует количеству распознаваемых классов. Но для уменьшения количества связей и вычислений для бинарного случая можно использовать один нейрон и при использовании в качестве функции активации гиперболический тангенс, выход нейрона со значением -1 означает не принадлежность к классу, напротив выход нейрона со значением 1 – означает принадлежность к классу [1-4].

Сверточные нейронные сети обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям. **Ядро** — представляет из себя фильтр, который скользит по всему изображению и находит признаки лица в любом его месте (инвариантность к смещениям).

Подвыборочный слой дает:

- увеличение скорости вычислений (минимум в 2 раза), за счет уменьшения размерности карт предыдущего слоя;
- фильтрация уже ненужных деталей;
- поиск признаков более высокого уровня (для следующего сверточного слоя).

Последние слои – слои обычного многослойного персептрона. Два полносвязных и один выходной. Этот слой отвечает за классификацию, с математической точки зрения моделирует сложную нелинейную функцию, оптимизируя которую улучшается качество распознавания.

1.2. Нечеткие множества

Основные понятия теории нечетких множеств

Определение 1.1. Нечеткие множества.

Пусть X универсальное множество, элементы которого обозначены через x . Принадлежность элементов в множестве A из X часто рассматривается как характеристическая функция μ_A из X в $\{0,1\}$, т.е. [1].

$$\mu_A(x) = \begin{cases} 1, & \text{если } x \in A, \\ 0, & \text{если } x \notin A. \end{cases}$$

Множество $\{0,1\}$ называется множеством оценок.

Если предположить, что не множество $\{0,1\}$, а действительный интервал $[0,1]$ является множеством оценок, тогда A будет нечетким множеством. В таком случае $\mu_A(x)$ будет называться функцией принадлежности. Чем больше $\mu_A(x)$ близко к 1, тем больше x принадлежит A [1].

Нечеткое множество A может характеризоваться множеством пар

$$A = \{(x, \mu_A(x)), x \in X\}.$$

В общем случае, нечетким множеством $A \subseteq X$ называется совокупность пар вида $(x, \mu_A(x))$, где $x \in X, \mu_A(x): X \rightarrow [0,1]$ (иногда $\mu_A(x): X \rightarrow L$ - структура типа решетки).

Нечеткое множество может быть конечным и бесконечным. Когда X -конечное множество, т.е. $X = \{x_1, \dots, x_n\}$, тогда нечеткое множество на X определяется следующим образом:

$$A = \mu_A(x_1)/x_1 + \dots + \mu_A(x_n)/x_n = \sum_{i=1}^n \mu_A(x_i)/x_i.$$

Когда X – бесконечное, тогда

$$A = \int_x \mu_A(x) / x.$$

Операции над нечеткими множествами

Пусть A и B – два нечетких множества в X с функциями принадлежности μ_A и μ_B , соответственно. Теоретико-множественные операции объединения, пересечения и дополнения для нечетких множеств определяются через их функции принадлежности следующим образом [1].

Определение 1.2. Объединение.

Объединение двух нечетких множеств A и B определяются следующим образом:

$$\forall x \in X, \mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\},$$

где $\mu_{A \cup B}$ - функция принадлежности для A и B .

Определение 1.3. Пересечение. Функция принадлежности $\mu_{A \cap B}$, определяется как

$$\forall x \in X, \mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}.$$

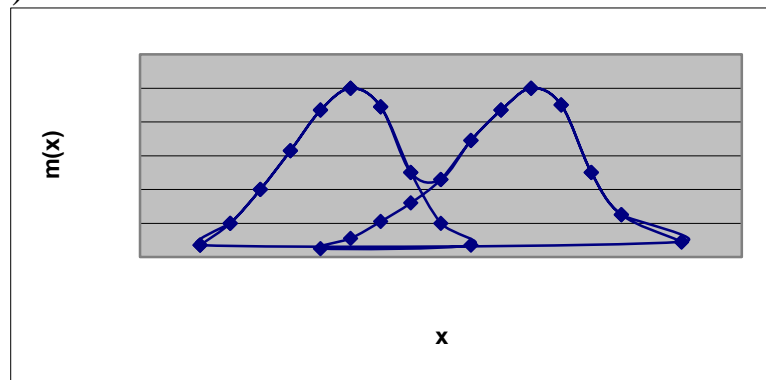
Пример.

$A = 0.07/2 + 0.2/3 + 0.4/4 + 0.63/5 + 0.87/6 + 1.0/7 + 0.89/8 + 0.5/9 + 0.2/10 + 0.07/11,$

$B = 0.05/6 + 0.11/7 + 0.21/8 + 0.32/9 + 0.46/10 + 0.69/11 + 0.87/12 + 1.0/13 + 0.9/14 + 0.5/15 + 0.25/16 + 0.09/18$

$A \cup B = 0.07/2 + 0.2/3 + 0.4/4 + 0.63/5 + 0.87/6 + 1.0/7 + 0.89/8 + 0.5/9 + 0.46/10 + 0.69/11 + 0.87/12 + 1.0/13 + 0.9/14 + 0.5/15 + 0.25/16 + 0.09/18$

(см. рис. 1.12).



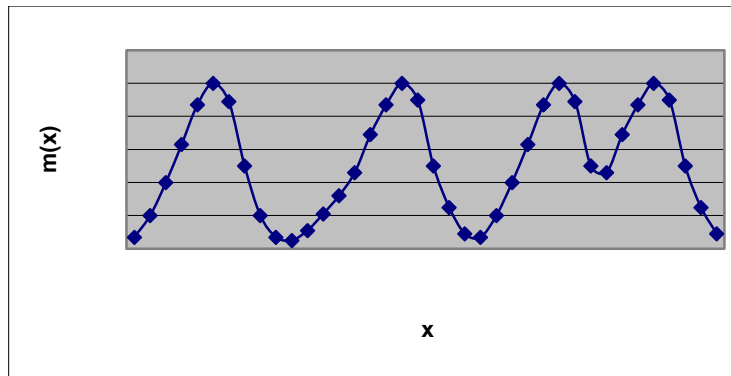


Рис.1.12. Объединение нечетких множеств

$A = 0.15/2 + 0.41/3 + 0.66/4 + 0.85/5 + 0.97/6 + 1/7 + 0.9/8 + 0.6/9 +$
 $+ 0.42/10 + 0.3/11 + 0.18/12 + 0.1/13 + 0.03/14,$
 $B = 0.05/5 + 0.1/6 + 0.16/7 + 0.25/8 + 0.35/9 + 0.47/10 + 0.62/11 +$
 $+ 0.8/12 + 0.94/13 + 1/14 + 0.97/15 + 0.83/16 + 0.5/17 + 0.2/18 + 0.07/19,$
 $A \cap B = 0.05/5 + 0.1/6 + 0.16/7 + 0.25/8 + 0.35/9 + 0.42/10 + 0.3/11 +$
 $+ 0.18/12 + 0.1/13 + 0.03/14$
 (см. рис. 1.13).

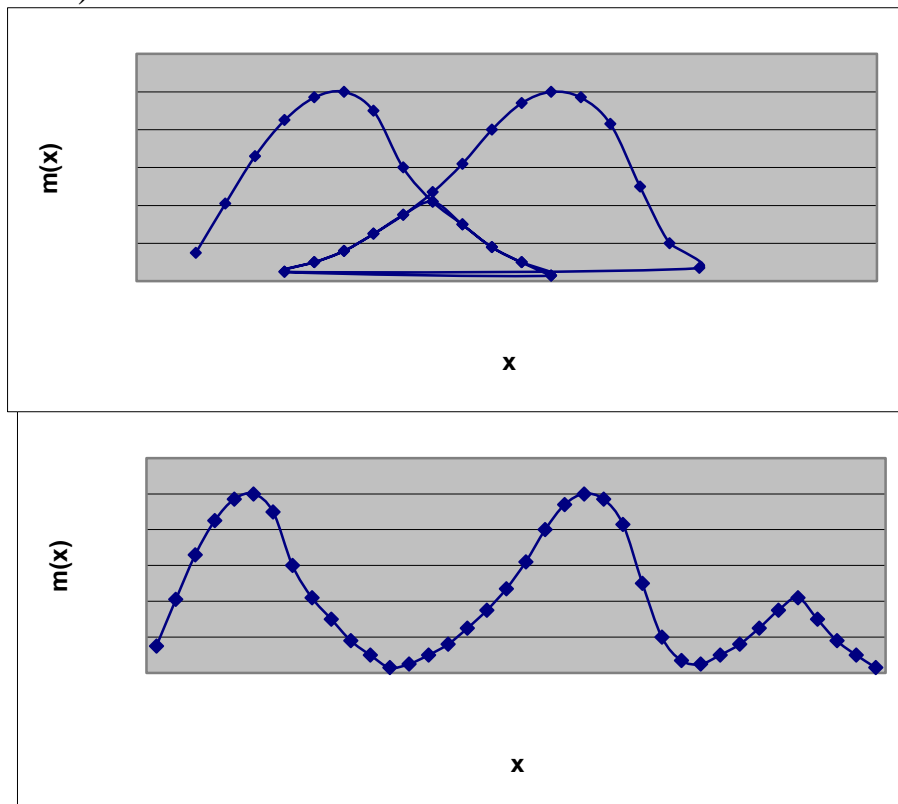


Рис.1.13. Пересечение нечетких множеств

Определение 1.4. Дополнение.

Дополнение \bar{A} множества A определяется следующим образом:

$$\forall x \in X, \mu_{\bar{A}}(x) = 1 - \mu_A(x).$$

Пример.

$$A = 0/1 + 0.05/2 + 0.14/3 + 0.27/4 + 0.5/5 + 0.76/6 + 0.93/7 + 1.0/8 + 0.96/9 + 0.84/10 + 0.62/11 + 0.37/12 + 0.25/13 + 0.16/14 + 0.09/15 + 0.03/16 + 0/17.$$

$$\bar{A} = 1.0/1 + 0.95/2 + 0.86/3 + 0.73/4 + 0.5/5 + 0.24/6 + 0.07/7 + 0/8 + 0.04/9 + 0.16/10 + 0.38/11 + 0.63/12 + 0.75/13 + 0.84/14 + 0.91/15 + 0.97/16 + 1.0/17.$$

(см. рис. 1.14).

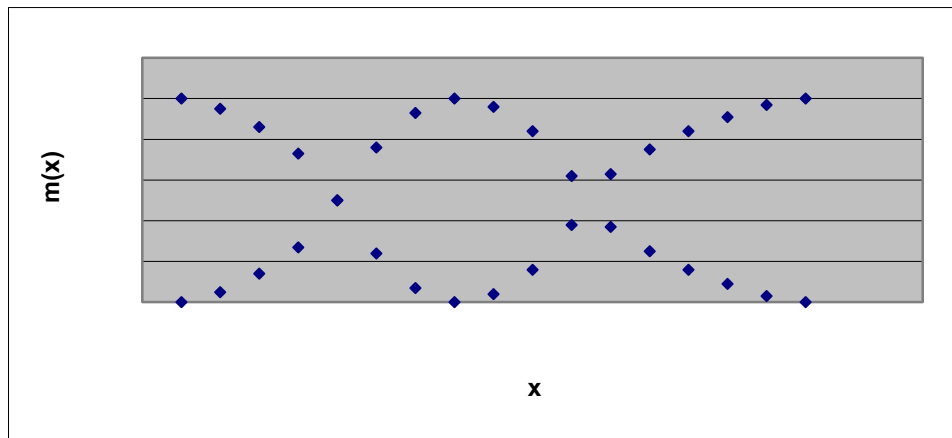
Для объединения и пересечения нечетких множеств можно использовать и другие операторы.

Алгебраическое произведение.

$$\forall x \in X, \mu_{A \cdot B}(x) = \mu_A(x) \cdot \mu_B(x).$$

Ограниченное произведение.

$$\forall x \in X, \mu_{A \otimes B}(x) = \max\{0, \mu_A(x) + \mu_B(x) - 1\}.$$



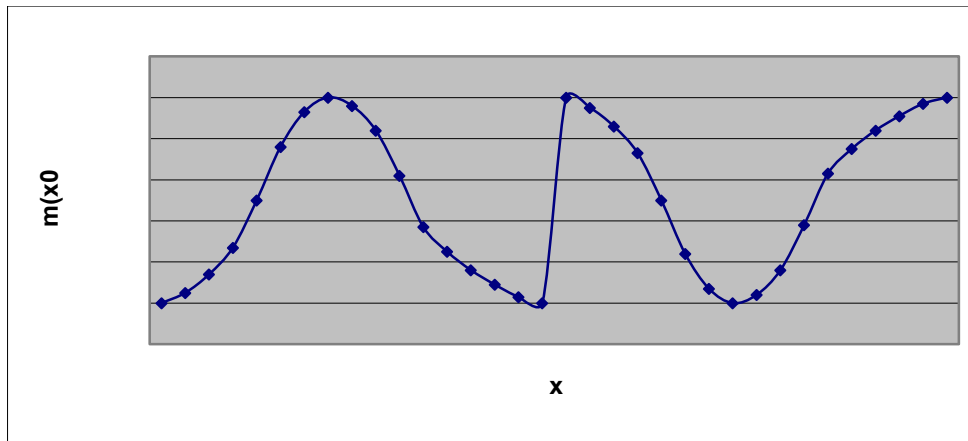


Рис.1.14. Дополнение нечеткого множества

Строгое (drastic) произведение.

$$\mu_{A \cap B}(x) = \begin{cases} \min\{\mu_A(x), \mu_B(x)\}, & \text{если } \mu_A(x) = 1 \text{ или } \mu_B(x) = 1 \\ 0, & \text{в остальных случаях} \end{cases}$$

Алгебраическая сумма.

$$\forall x \in X, \mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x).$$

Ограниченная сумма.

$$\forall x \in X, \mu_{A \cup B}(x) = \min\{1, \mu_A(x) + \mu_B(x)\}.$$

Строгая (drastic) сумма.

$$\mu_{A \cup B}(x) = \begin{cases} \max\{\mu_A(x), \mu_B(x)\}, & \text{если } \mu_A(x) = 0 \text{ или } \mu_B(x) = 0 \\ 0, & \text{в остальных случаях} \end{cases}$$

Пример. Алгебраическое произведение нечетких множеств А и В.

$$A = 0.1/1 + 0.24/2 + 0.4/3 + 0.63/4 + 0.82/5 + 0.94/6 + 1.0/7 + 0.98/8 + 0.91/9 + 0.76/10 + 0.57/11 + 0.35/12 + 0.2/13 + 0.1/14 + 0.04/15,$$

$$B = 0.02/4 + 0.09/5 + 0.2/6 + 0.32/7 + 0.46/8 + 0.61/9 + 0.76/10 + 0.88/11 + 0.96/12 + 1.0/13 + 0.96/14 + 0.85/15 + 0.62/16 + 0.37/17 + 0.2/18 + 0.09/19$$

$$A * B = 0/3 + 0.01/4 + 0.07/5 + 0.19/6 + 0.32/7 + 0.45/8 + 0.55/9 + 0.58/10 + 0.5/11 + 0.34/12 + 0.2/13 + 0.96/14 + 0.03/15 + 0/16.$$

(см. рис. 1.15).

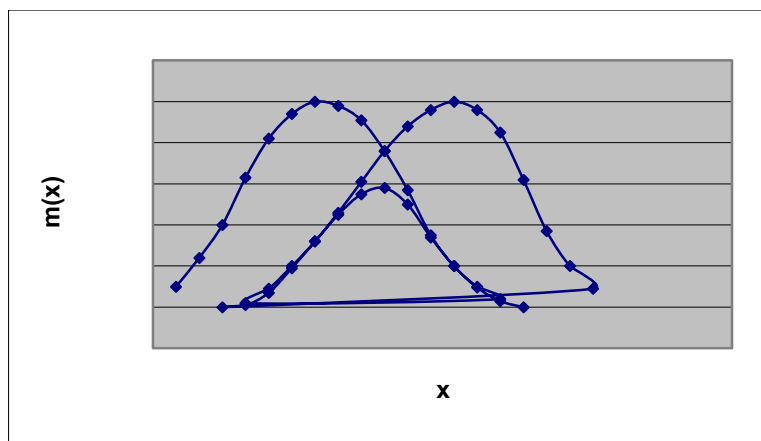


Рис.1.15.Алгебраическое произведение нечетких множеств

Алгебраическая сумма нечетких множеств А и В.

$$A=0.03/1+0.1/2+0.28/3+0.52/4+0.75/5+0.94/6+1/7+0.96/8+0.87/9$$

$$+$$

$$+0.71/10+0.55/11+0.4/12+0.28/13+0.19/14+0.12/15+0.06/16+0.02/17,$$

$$B=0/1+0/2+0/3+0.02/4+0.06/5+0.12/6+0.17/7+0.25/8+0.35/9+0.5/$$

$$10+$$

$$+0.68/11+0.82/12+0.95/13+1/14+0.95/15+0.62/16+0.35/17+$$

$$+0.17/18+0.06/19,$$

$$A \hat{+} B =0.03/1+0.1/2+0.28/3+0.52/4+0.75/5+0.94/6+1.0/7+0.96/8+$$

$$+0.91/9+0.86/10+0.86/11+0.88/12+0.96/13+1.0/14+0.95/15+$$

$$+0.62/16+0.35/17+0.17/18+0.06/19.$$

(см. рис. 1.16).

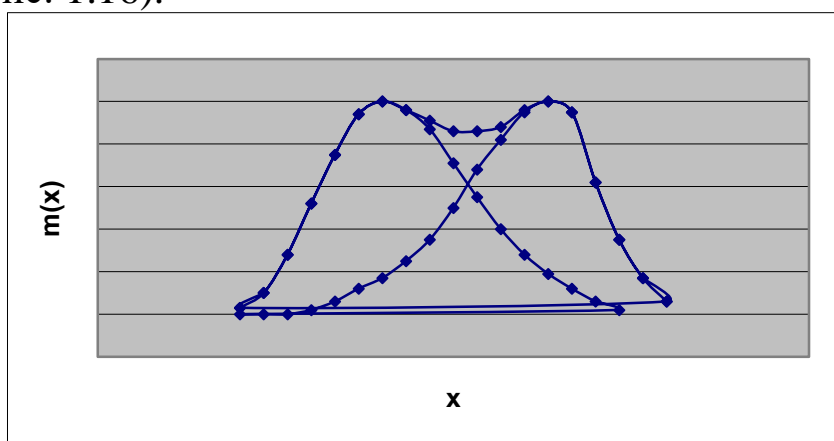


Рис. 1.16.Алгебраическая сумма нечетких множеств

Ограниченная сумма нечетких множеств А и В.

$$\begin{aligned}
 A &= 0.06/1 + 0.17/2 + 0.31/3 + 0.5/4 + 0.67/5 + 0.82/6 + 0.93/7 + 1.0/8 + 0.98/9 + \\
 &+ 0.89/10 + 0.75/11 + 0.6/12 + 0.45/13 + 0.33/14 + 0.23/15 + 0.14/16 + \\
 &+ 0.08/17 + 0.03/18, \\
 B &= 0.03/4 + 0.08/5 + 0.15/6 + 0.26/7 + 0.4/8 + 0.55/9 + 0.7/10 + 0.85/11 + \\
 &+ 0.95/12 + 1.0/13 + 0.96/14 + 0.85/15 + 0.6/16 + 0.33/17 + 0.18/18 + 0.09/19, \\
 A \cup B &= 0.06/1 + 0.17/2 + 0.31/3 + 0.53/4 + 0.75/5 + 0.97/6 + 1.0/7 + 1.0/8 + 1.0/9 + \\
 &+ 1.0/10 + 1.0/11 + 1.0/12 + 1.0/13 + 1.0/14 + 1.0/15 + 0.64/16 + 0.41/17 + \\
 &+ 0.21/18 + 0.09/19.
 \end{aligned}$$

(см. рис. 1.17).

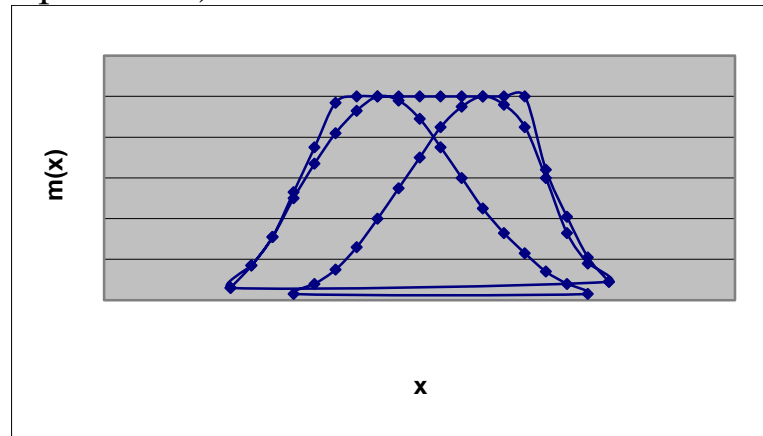


Рис.1.17. Ограниченная сумма нечетких множеств А и В.

Ограниченное произведение нечетких множеств А и В.

$$\begin{aligned}
 A &= 0.03/1 + 0.15/2 + 0.5/3 + 0.77/4 + 0.93/5 + 1.0/6 + 0.96/7 + 0.85/8 + 0.71/9 + \\
 &+ 0.55/10 + 0.4/11 + 0.27/12 + 0.18/13 + 0.11/14 + 0.05/15 + 0.01/16, \\
 B &= 0.04/5 + 0.1/6 + 0.17/7 + 0.28/8 + 0.4/9 + 0.55/10 + 0.71/11 + 0.89/12 + 0.98/13 + \\
 &+ 1.0/14 + 0.93/15 + 0.65/16 + 0.2/17 + 0.06/18 + 0.01/19,
 \end{aligned}$$

$$\begin{aligned}
& A \cap B \\
& = 0/1 + 0/2 + 0/3 + 0/4 + 0/5 + 0.1/6 + 0.13/7 + 0.13/8 + 0.11/9 + 0.1/10 + 0.11/ \\
& 11 + 0.16/12 + 0.16/13 + 0.11/14 + 0/15 + 0/16 + 0/17 + 0/18 + 0/19. \\
& \text{(см. рис. 1.18)}
\end{aligned}$$

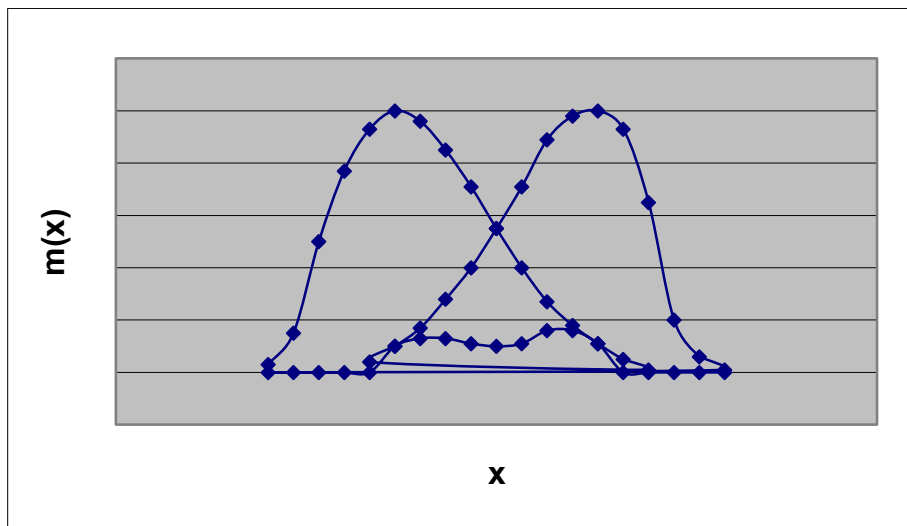


Рис.1.18. Ограниченное произведение нечетких множеств А и В.

Определение 1.5. Ограниченная и симметрическая разности.

Ограниченная разность $|_-$ нечетких множеств определяется формулой

$$\forall x \in X, \mu_{A|_B}(x) = \max(0, \mu_A(x) - \mu_B(x)).$$

$A|_B$ есть нечеткое множество элементы которого принадлежат А больше чем В.

Симметрическая разность нечетких множеств – это нечеткое множество $A \nabla B$ элементов, которые больше принадлежат А нежели В:

$$\forall x \in X, \mu_{A \nabla B}(x) = |\mu_A(x) - \mu_B(x)|.$$

Примеры ограниченной и симметрической разности нечетких множеств А и В:

$$\begin{aligned}
A = & 0.08/1 + 0.23/2 + 0.45/3 + 0.7/4 + 0.86/5 + 0.96/6 + 1/0/7 + 0.98/8 + \\
& + 0.92/9 + 0.82/10 + 0.67/11 + 0.47/12 + 0.3/13 + 0.13/14,
\end{aligned}$$

$$\begin{aligned}
B = & 0.03/6 + 0.08/7 + 0.18/8 + 0.34/9 + 0.55/10 + 0.7/11 + 0.84/12 + 0.94/13
\end{aligned}$$

+

$$+0.99/14+1.0/15+0.96/16+0.82/17+0.6/18+0.2/19,$$

$$A \setminus B = 0.08/1+0.23/2+0.45/3+0.7/4+0.86/5+0.93/6+0.92/7+0.8/8+ \\ +0.58/9+0.27/10+0/11,$$

$$A \nabla B \\ = 0.08/1+0.23/2+0.45/3+0.7/4+0.86/5+0.96/6+1.0/7+0.98/8+ \\ 0.92/9+0.82/10+0.03/11+0.36/12+0.65/13+0.86/14+1.0/15+ \\ 0.96/16+0.82/17+0.6/18+0.2/19.$$

(см. рис.1.19.а и 1.19.б).

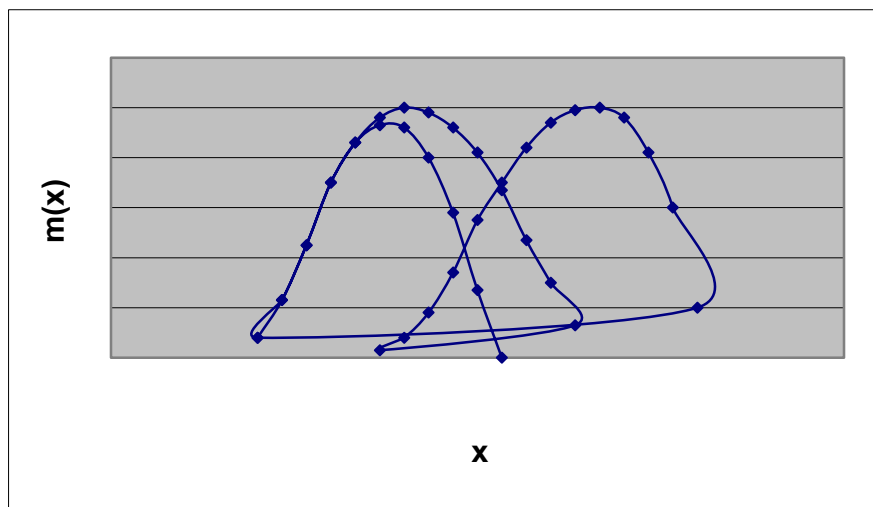


Рис.1.19.а. Ограниченная разность

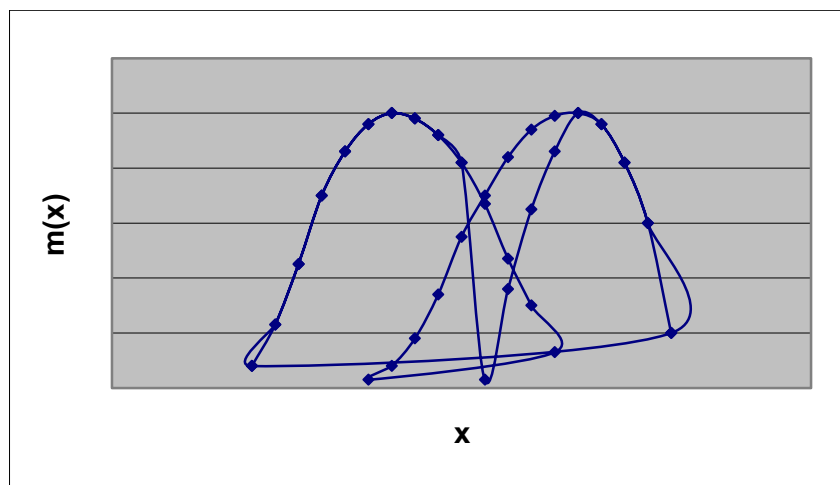


Рис.1.19.б. Симметрическая разность

Определение 1.6. m -я степень нечеткого множества A равна:

$$\mu_{A^m}(x) = [\mu_A(x)]^m, \quad \forall x \in X, \forall m \in R^+,$$

где R^+ - положительно определенное множество действительных чисел.

Определение 1.7. Концентрация, растяжение нечетких множеств. Пусть A является нечетким множеством на универсуме:

$$A = \{(x : \mu_A(x)) \mid x \in X\}.$$

Тогда, возникающие посредством возведения в степень s помощью оператора концентрирования Con_s нечеткие множества

$$Con_s A = \{(x : (\mu_A(x))^s) \mid x \in X\}$$

называются концентрациями A , а посредством извлечения корня s помощью оператора расширения $dil_s A = \{(x : \sqrt[s]{\mu_A(x)}) \mid x \in X\}$ - называются расширениями A .

Следствие. Так как выражение $[\mu_A(x)]^n \leq \mu_A(x) \leq \sqrt[n]{\mu_A(x)}$ действительно для всех $x \in X$ и $n > 1$, то и соотношение подмножеств $Con_n A \subset A \subset dil_n A$ также является действительным.

Концентрация нечеткого множества: $n=2$.

$A=0.03/1+0.1/2+0.21/3+0.37/4+0.57/5+0.8/6$
 $+0.96/7+1.0/8+0.94/9+$
 $+0.7/10+ 0.42/11+0.27/12+0.17/13+0.09/14+0.03/15,$
 $A^2=0.0009/1+0.01/2+0.044/3+0.137/4+0.325/5+0.64/6+0.92/7+1.0/8+$
 $+0.884/9+0.49/10+0.174/11+0.07/12+0/03/13+0/01/14+0/0009/15$
 (см. рис. 1.20).

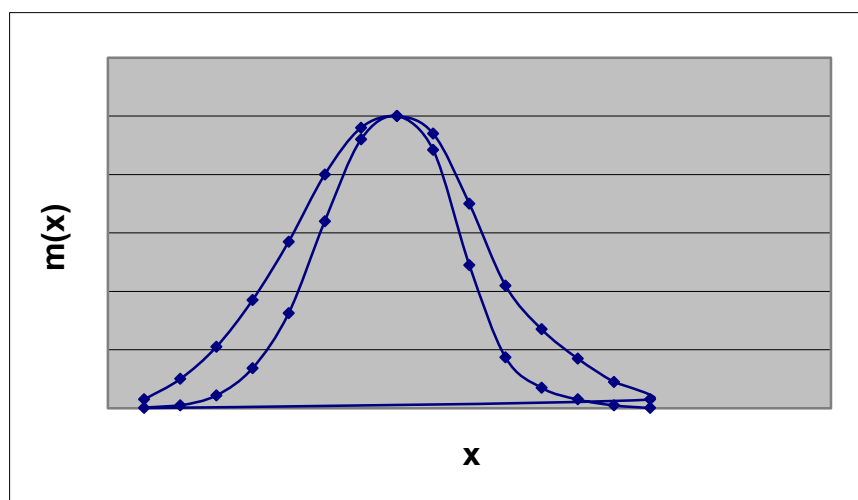


Рис.1.20. Концентрация нечетких множеств

Растяжение нечеткого множества $n=2$.

$$A=0.03/2+0.06/3+0.13/4+0.23/5+0.4/6+0.61/7+0.82/8+0.96/9+ \\ +1.0/10+0.94/11+0.74/12+0.51/13+0.33/14+0.23/15+0.16/16+0.1/17+ \\ +0.05/18+0.02/19,$$

$$A^{1/2} \\ =0.17/2+0.25/3+0.36/4+0.48/5+0.63/6+0.78/7+0.9/8+0.98/9+1.0/10+ \\ +0.97/11+0.86/12+0.72/13+0.57/14+0.48/15+0.4/16+0.3/17+ \\ +0.22/18+0.15/19$$

(см. рис. 3.15).

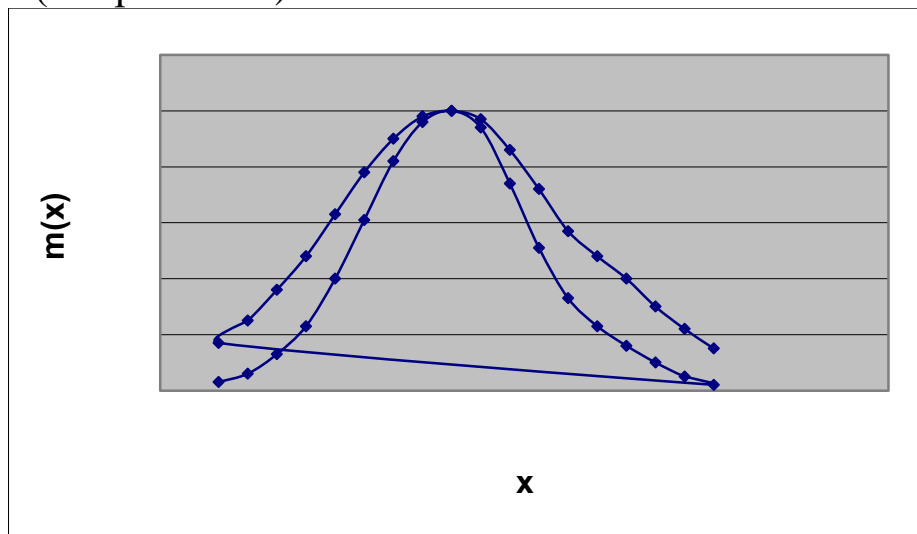


Рис.1.21. Растяжение нечетких множеств

Примеры преобразования нечетких множеств с использованием операций концентрации и растяжения приведены ниже [6].

Таблица 1.2

A	$= \int \mu_A(x) / x$
Очень A	$= \int [\mu_A(x)]^2 / x$
Очень очень A	$= \int [\mu_A(x)]^4 / x$
Более или менее A	$= \int \sqrt{\mu_A(x)} / x$
Немного A	$= \int \sqrt[4]{\mu_A(x)} / x$
Не A	$= \int (1 - \mu_A(x)) / x$
Не очень A	$= \int (1 - [\mu_A(x)]^2) / x$

Определение 1.8. Нечеткие точки, нечеткие интервалы, нечеткие области.

Нечеткая точка есть выпуклое нечеткое подмножество действительной прямой \mathbb{R} .

Как показано в [7], нечеткие точки представляются симметричными интервалами для компонентов, отражающих эту неточность (в случае гиперпирамидального представления).

В случае эллиптического гиперпараболоидального представления неточность учитывается относительно всех направлений пространства посредством матрицы, которая играет роль аналогичную ковариационной матрице в математической статистике для случая одной особо наблюдаемой точки x_0 .

Если границами интервала являются нормальные выпуклые нечеткие множества, то он называется нечетким интервалом.

Нечеткие интервалы могут определяться либо с помощью выбора четкого интервала для формирования ядра, от которого функция принадлежности уменьшается до нуля, или посредством выбора двух нечетких чисел в качестве концов интервала. Вообще мы можем построить нечеткий регион в пространстве \mathbb{R}^k посредством выбора четкого региона, окруженного нечеткой переходной зоной, в которой функции принадлежности уменьшается до нуля монотонно. Альтернативный способ представления нечеткой области – это определение нечеткой гиперповерхности, формирующей его границу. Такая нечеткая гиперповерхность имеет четкую гиперповерхность свонго ядра, при удалении от которого значения функции принадлежности монотонно убывают во всех направлениях.

Нечеткие отношения и нечеткие ограничения

Термин «отношение» используется для обозначения некоторых типов отображений, которые задаются на одном и том же универсуме X . В подобном случае отображение $\Gamma: X \rightarrow X$ это отображение из множества X в себя, которое определяется парой $\{X, \Gamma\}$, где $\Gamma \subseteq X^2$.

Поскольку элементами множества X^2 являются упорядочивающие пары, можно сказать, что отношение – это множество упорядочивающих пар. Т.к. каждая пара соединяет

только 2 элемента множества X^2 между собой. Такое отношение называется бинарным. Если элементами множества X^n являются упорядочивающие n-арные пары, такое отношение называется n-арным отношением. Частный случай – тернарное отношение – множество из упорядочивающих троек.

Понятие нечеткое отношение – это обобщение четких (как было отмечено выше) отношений в теории нечетких множеств. Оно может моделировать ситуации, где взаимодействия между элементами являются более или менее сильными.

Можно различать различные типы отношений. Например, отношение эквивалентности порядка, превосходства и т.д.

Определение 1.9. Нечеткое отношение [1,2].

Пусть X_1, X_2, \dots, X_n - n-универсумов. n-арное нечеткое отношение R в X_1, X_2, \dots, X_n является множеством на X_1, X_2, \dots, X_n и выражается как:

$$R_{X_1 \times \dots \times X_n} = \{((X_1, \dots, X_n), (X_1, \dots, X_n)) \mid (X_1, \dots, X_n) \in X_1 \times \dots \times X_n\}.$$

Как отмечалось выше, обычное отношение – частный случай нечетких отношений. Носителем нечеткого отношения R на множестве X называют подмножество декартова произведения $X \times X$ вида

$$\text{sup } pR = \{(u, v) \mid (u, v) \in X \times X, \mu_R(u, v) > 0\}.$$

Носитель нечеткого отношения следует понимать как отношение на множестве X , связывающее все пары (u, v) , для которых степень выполнения данного нечеткого отношения не равна нулю.

По аналогии с нечеткими множествами определяется и множество уровня нечеткого отношения, т.е.

$$R_\alpha = \{(u, v) \mid (u, v) \in X \times X, \mu_R(u, v) \geq \alpha\}.$$

Перейдем к рассмотрению операций над нечеткими отношениями, некоторые из которых являются аналогами операций над нечеткими множествами. А некоторые присущи только нечетким отношениям.

Перейдем к рассмотрению операций над нечеткими отношениями, некоторые из которых являются аналогами операций над нечеткими множествами, а некоторые присущи только нечетким отношениям.

Пересечением нечетких отношений P и Q на $X \times X$ называют нечеткое отношение $P \cap Q$ определяемое функцией принадлежности

$$\mu_{P \cap Q}(u, v) = \mu_P(u, v) \wedge \mu_Q(u, v) = \min\{\mu_P(u, v), \mu_Q(u, v)\}, \\ \forall u, v \in X \times X.$$

Объединением нечетких отношений P и Q на $X \times X$ называют нечеткое бинарное отношение $P \cup Q$ определяемое функцией принадлежности

$$\mu_{P \cup Q}(u, v) = \mu_P(u, v) \vee \mu_Q(u, v) = \max\{\mu_P(u, v), \mu_Q(u, v)\}, \\ \forall u, v \in X \times X.$$

Дополнением нечеткого отношения $R \subseteq X \times X$ называют отношение \bar{R} с функцией принадлежности

$$\mu_{\bar{R}}(u, v) = 1 - \mu_R(u, v), \quad \forall u, v \in X \times X.$$

Обратным отношением к отношению R называют отношение R^{-1} с функцией принадлежности

$$\mu_{R^{-1}}(u, v) = \mu_R(v, u), \quad \forall u, v \in X \times X.$$

Очевидно, что матрица R^{-1} является транспонированной к матрице R .

Определение 1.10. Нечеткое ограничение.

Пусть $v = (v_1, \dots, v_n)$ - переменная на $X = X_1 \times \dots \times X_n$.

Нечеткое ограничение, обозначаемое $R(v)$, есть нечеткое отношение R , которое действует как гибкое ограничение на значения переменной v . В данном контексте переменная может рассматриваться как тройка $(v, x, R(v))$; v - имя переменной.

Проекцией нечеткого отношения R на $X_{i_1} \dots X_{i_k}$, (i_1, \dots, i_k) -подпоследовательность $(1, 2, \dots, n)$, является отношение на $X_{i_1} \times \dots \times X_{i_k}$, определяемое как

$$proj(R; X_{i_1}, \dots, X_{i_k}) = \int_{X_{i_1} \times \dots \times X_{i_k}} \sup_{X_{j_1}, \dots, X_{j_k}} \mu_R(X_1, \dots, X_n) / (X_{i_1}, \dots, X_{i_k}),$$

где (j_1, \dots, j_k) - подпоследовательность дополняющаяся до (i_1, \dots, i_k) в $(1, \dots, n)$.

Проекции называются также маргинальными нечеткими ограничениями. Обратное, если R - нечеткое множество в $X_{i_1} \times \dots \times X_{i_k}$, тогда цилиндрическое расширение в $X_1 \times \dots \times X_n$ - это нечеткое множество $C(R)$ на $X_1 \times \dots \times X_n$, определяемое посредством

$$C(R) = \int_{X_1 \times \dots \times X_n} \mu_R(X_{i_1}, \dots, X_{i_k}) / (X_1, \dots, X_n).$$

n -арное нечеткое ограничение $R(v_1, \dots, v_n)$ называется разделимым если и только если

$$R(v_1, \dots, v_n) = R(v_1) \times \dots \times R(v_n),$$

где \times обозначает картезианское произведение и $R(v_i)$ - проекция R на X , т.е.

$$\mu_R(X_1, \dots, X_n) = \min_{i=1, n} \mu_{proj[R; X_i]}(X_i).$$

В терминах цилиндрического расширения эта формула может быть переписана как

$$R = \bigcap_{i=1, n} C(proj[R; X_i]).$$

R разделимо, если и только если оно является соединением ее проекций. Если R разделимо, то все ее маргинальные нечеткие ограничения также разделимы. Переменные v_1, \dots, v_n называются не взаимодействующими, если их ограничение $R(v_1, \dots, v_n)$ является разделимым нечетким ограничением.

Пример.

Вычислим нечеткое отношение $R = A \times B$, если

$$A = 0,1/4 + 0,3/5 + 0,4/6,$$

$$B = 0,33/10 + 0,45/11 + 0,78/12.$$

Вместо **min** операции будем использовать **max** и **prod** операции

$$\mu_R = \max(\mu_A, \mu_B).$$

$$R = A \times B = 0,33/(4.10) + 0,45/(4.11) + 0,78/(4.12) + 0,33/(5.10) + 0,45/(5.11) + 0,78/(5.12) + 0,4/(6.10) + 0,45/(6.11) + 0,78/(6.12).$$

$$R = \begin{vmatrix} 0,33 & 0,45 & 0,78 \\ 0,33 & 0,45 & 0,78 \\ 0,4 & 0,45 & 0,78 \end{vmatrix}$$

$$\mu_R = (\mu_A \times \mu_B).$$

$$R = A \times B = 0,033/(4.10) + 0,045/(4.11) + 0,078/(4.12) + 0,099/(5.10) + 0,135/(5.11) + 0,234/(5.12) + 0,132/(6.10) + 0,180/(6.11) + 0,312/(6.12).$$

$$R = \begin{vmatrix} 0,033 & 0,045 & 0,078 \\ 0,099 & 0,135 & 0,234 \\ 0,132 & 0,180 & 0,312 \end{vmatrix}$$

Бинарные нечеткие отношения

Бинарное нечеткое отношение – это обобщение классического бинарного отношения.

Бинарное отношение R в $X \times Y$ - это нечеткое множество в $X \times Y$. Пусть R – бинарное нечеткое отношение на $X \times Y$. Домен отношения R , обозначаемый $\text{dom}(R)$, и его ранг – $\text{ran}(R)$ определяются, соответственно:

$$\mu_{\text{dom}(R)}(x) \sup_y \mu_R(x, y), \forall x \in X,$$

$$\mu_{\text{ran}(R)}(y) \sup_x \mu_R(x, y), \forall y \in Y.$$

Определение 1.11. Sup-Starкомпозиция: Если R и S являются нечеткими отношениями в $U \times V$ и $V \times W$, соответственно, композиция R и S является нечетким отношением, обозначаемым $R \circ S$ и определяется как:

$$R \circ S = \{[(x, y), \sup_{y \in Y} (\mu_R(x, y) * \mu_S(y, z))] \mid x \in X, y \in Y, z \in Z\},$$

где $*$ может являться любым оператором из класса треугольных норм, а именно: минимум, алгебраическое произведение, ограниченное произведение или строгое (drastic) произведение [3].

Уравнение может интерпретироваться следующим образом: $\mu_{R \circ S}(x, z)$ - это сила множества цепей, соединяющих X с Z . Каждая цепь имеет форму x - y - z . Сила такой цепи равна силе наислабейшего соединения. Сила отношения между x и z является силой наисильнейшего соединения между x и z .

Пусть A – нечеткое множество в X . Тогда:

$$\mu_{A \circ R}(y) = \sup_x \min(\mu_A(x), \mu_R(x, y)).$$

Мы говорим, что $B = A \circ R$ является нечетким множеством, индуцированным из A через R . Эта индукция обобщает хорошо известное четкое правило: если $x=a$ и $y=f(x)$, то $y=f(a)$.

Имеем $B = \text{proj}[C(A) \cap R; Y]$.

Нечеткое отношение можно представить на конечном универсуме.

Когда связанные универсумы X и Y конечны, нечеткое отношение R на $X \times Y$ может быть представлено в виде матрицы $[R]$, терм которого $[R]_{ij}$ является

$$\mu_R(x_i, y_j) = r_{ij}, \quad i = \overline{1, n}; j = \overline{1, m},$$

где $|X| = n$ $|Y| = m$.

Композиция конечных нечетких отношений с учетом

$$[S]_{jk} = S_{jk}, \quad k = \overline{1, p}; \quad P = |Z|,$$

может рассматриваться как матричное произведение

$$[ROS]_{ik} = \sum_j r_{ij} S_{jk},$$

где суммирование выполняется операцией \max , а умножение операцией \min .

Так как $R \circ S$ может быть записано в виде:

$$\text{proj}[C(R) \cap C(S); X \times Z],$$

где R и S заданы на $X \times Y$ и $Y \times Z$, соответственно, то другие композиции могут быть введены посредством модификации оператора, использованного для пересечения.

Изменяя \min на $*$, мы вводим $R * S$ через

$$\mu_{R*S}(x, z) = \sup_y (\mu_R(x, y) * \mu_S(y, z)).$$

Мы можем встретить иной тип альтернативных композиций, $\inf\text{-max}$, $\sup\text{-prod}$ и другие.

Нечеткое отношение ЕСЛИ-ТО

Пусть A и B – нечеткие подмножества на универсумах X и Y .

Для связи нечетких подмножеств A и B , заданных на разных областях рассуждений X и Y , вводится понятие нечеткого условного утверждения (лингвистической импликации), т.е.

$$A \rightarrow B \quad \text{при «ЕСЛИ } A \text{ ТО } B\text{»}.$$

Полученное импликацией отношение R выражается в терминах картезианского произведения подмножеств A и B , обозначается как $R = A \times B$, и его функция принадлежности определяется как [1].

$$\mu_R(x, y) = \mu_{A \times B}(x, y) = \min[\mu_A(x), \mu_B(y)], \quad x \in X, \quad y \in Y.$$

Пример

1. Задана нечеткая импликация: если A то B . Вычислить нечеткое отношение $R = A \times B$, с использованием \min операции композиции, если

$$\text{а) } A = 0.1/20 + 0.3/21 + 0.4/22,$$

$$B = 0.33/60 + 0.45/65 + 0.78/70;$$

$$R = A \times B = 0.1/(20,60) + 0.1/(20,65) + 0.1/(20,70) + 0.3/(21,60) + 0.3/(21,65) + 0.3/(21,70) + 0.33/(22,60) + 0.4/(22,65) + 0.4/(22,70).$$

Может встретиться также вложенное нечеткое отношение. В таких случаях нечеткое условное утверждение является

вложенным, т.е. имеет форму ЕСЛИ А ТО ЕСЛИ В ТО С. Тогда нечеткое отношение R записывается в виде:

$$R = A \times (B \times C) = A \times B \times C$$

Нечеткая импликация может состоять из двух импликаций. Эти простые импликации соединяются с использованием соединителей «или» («иначе»), «и» и т.д.

Пример.

Пусть дана импликация

ЕСЛИ A_1 ТО B_1

или (иначе) ЕСЛИ A_2 ТО B_2

где A_1, A_2 - нечеткие подмножества на X, а B_1, B_2 - нечеткие подмножества на Y.

Результирующее нечеткое отношение R вычисляется как объединение индивидуальных нечетких отношений R_i ($i = 1, 2$):

$$R = \bigcup_{i=1,2} R_i = \bigcup_{i=1,2} A_i \times B_i$$

Функция принадлежности R определяется как

$$\mu_R(x, y) = \max_x \{ \min[\mu_{A_1}(x), \mu_{B_1}(y)], \min[\mu_{A_2}(x), \mu_{B_2}(y)] \}.$$

Это может быть расширено на случай, когда имеем дело не с одной, а более чем двумя импликациями.

Пусть заданы нечеткое отношение $R = X \times Y$ и значение A' нечеткого подмножества A. Отношение используется для вывода соответствующего значения B' посредством применения композиционного правила вывода, которое записывается как

$$B' = A' \circ R = A' \circ (A \times B).$$

Функция принадлежности определяется как

$$\mu_{B'}(y) = \max_x \min[\mu_{A'}(x), \mu_R(x, y)].$$

В случае тернарного нечеткого отношения формулы имеют вид:

$$C' = A' \circ (B' \circ R) = A' \circ (B' \circ (A \times B \times C)),$$

$$\mu_{C'}(z) = \max_x \min \left[\mu_{A'}(x), \max_y \min[\mu_B(y), \mu_R(x, y, z)] \right].$$

Пример.

Задана нечеткая импликация ЕСЛИ А ТО ЕСЛИ В ТО С. Вычислим нечеткое отношение $R = A \times B \times C$.

$$A = 0.3/5 + 0.5/6 + 0.8/7,$$

$$B = 0.3/15 + 0.5/16 + 0.8/17,$$

$$\begin{aligned}
C &= 0.2/25 + 0.4/26 + 0.6/27; \\
R &= A \times B \times C = \\
&= 0.2/(5,15,25) + 0.3/(5,15,26) + 0.3/(5,15,27) + \\
&+ 0.2/(5,16,25) + 0.3/(5,16,26) + 0.3/(5,16,27) + \\
&+ 0.2/(5,17,25) + 0.3/(5,17,26) + 0.3/(5,17,27) + \\
&+ 0.2/(6,15,25) + 0.4/(6,15,26) + 0.3/(6,15,27) + \\
&+ 0.2/(6,16,25) + 0.4/(6,16,26) + 0.5/(6,16,27) + \\
&+ 0.2/(6,17,25) + 0.4/(6,17,26) + 0.5/(6,17,27) + \\
&+ 0.2/(7,15,25) + 0.3/(7,15,26) + 0.3/(7,15,27) + \\
&+ 0.2/(7,16,25) + 0.4/(7,16,26) + 0.6/(7,16,27) + \\
&+ 0.2/(7,17,25) + 0.4/(7,17,26) + 0.56/(7,17,27).
\end{aligned}$$

Определение 1.12. Нечеткий граф.

С понятием нечеткого отношения вплотную связано понятие нечеткого графа. Допустим, что E обычное множество узлов. Нечеткий граф определяется как [3]

$$G(X_i, X_j) = \{((X_i, X_j), \mu_G(X_i, X_j)) / (X_i, X_j) \in E \times E\}.$$

Если E – нечеткое множество, то нечеткий граф будет определяться аналогично нечетким отношениям.

Пример.

$E = \{X_1, X_2, X_3, X_4\}$. Тогда нечеткий граф может быть представлен как

$$\begin{aligned}
G(X_i, X_j) &= \{[(X_1, X_2), 0.3], [(X_1, X_3), 0.6], [(X_1, X_1), 1], [(X_2, X_1), 0.4], \\
&[(X_3, X_1), 0.2], [(X_3, X_2), 0.5], [(X_4, X_3), 0.8]\}.
\end{aligned}$$

Нечеткая арифметика

Нечеткие числа и операции с нечеткими числами

Сначала рассмотрим основные определения нечетких чисел и операций над ними [1-4].

Определение 1.13. Нечеткое число. Нечеткое число A на действительной прямой – это нечеткий набор, характеризуемый функцией принадлежности $\mu_A : R \rightarrow [0,1]$. Нечеткое число A может быть представлено как

$$A = \int_R \mu_A(x) / x,$$

где \int - знак объединения по всем $x \in R$.

Определение 1.14. Выпуклость нечеткого числа. Нечеткое число A на действительной прямой выпукло, если для реальных чисел $x, y, z \in R$ $x \leq y \leq z$

$$\mu_A(y) \geq \min(\mu_A(x), \mu_A(z)).$$

Определение 1.15. Нормальность нечеткого числа. Нечеткое число A на действительной оси называется нормальным, если $\max_x \mu_A(x) = 1$.

Определение 1.16. Границы нечеткого числа. Если для числа a выполняется соотношение

$$\forall \delta \mu_A = 0; \quad \mu(a - \delta) \neq 0, \mu(a + \delta) \neq 0,$$

то оно называется границей функции принадлежности. Если учесть, что таких границ две: верхняя (b) и нижняя (a), нечеткое число A можно записать как

$$A = \int_a^{\bar{a}} (x - a) / x + \int_{\bar{a}}^b (b - x) / x.$$

Принцип обобщения. Пусть заданы нечеткие числа A и B на действительной прямой R . Операцию $*$ можно выполнить над A и B используя соотношение

$$A * B = \int_R \min(\mu_A(x), \mu_B(y)) / (x * y).$$

Используя взамен гипотетической операции $*$ арифметические $+$, $-$, \times , $:$, можно получить четыре арифметические действия над A и B :

$$\begin{aligned}
A + B &= \int_R \min(\mu_A(x), \mu_B(y)) / (x + y) . \\
A - B &= \int_R \min(\mu_A(x), \mu_B(y)) / (x - y) . \\
A \times B &= \int_R \min(\mu_A(x), \mu_B(y)) / (x \times y) . \\
A : B &= \int_R \min(\mu_A(x), \mu_B(y)) / (x : y) .
\end{aligned}$$

Можно получить

$$\begin{aligned}
A * B &= \left(\int_a^{\bar{a}} \mu_A(x) / x + \int_{\bar{a}}^b \mu_A(x) / x \right) * \left(\int_a^{\bar{b}} \mu_B(x) / x + \int_{\bar{b}}^{b'} \mu_B(x) / x \right) = \\
&= \int_{a''}^{\bar{a} * \bar{b}} \mu_{A * B}(x) / x + \int_{\bar{a} * \bar{b}}^{b''} \mu_{A * B}(x) / x,
\end{aligned}$$

где a'', b'' получаются из a, b, a', b' , в зависимости от конкретной операции, $\mu_{A * B}(x)$ определяется в зависимости от конкретной операции и нормировки μ . Вычислим $A + B$

$$\begin{aligned}
A + B &= \left(\int_a^{\bar{a}} \mu_A(x) / x + \int_{\bar{a}}^b \mu_A(x) / x \right) + \left(\int_a^{\bar{b}} \mu_B(x) / x + \int_{\bar{b}}^{b'} \mu_B(x) / x \right) = \\
&= \int_{a''}^{\bar{c}} \mu_C(x) / x + \int_{\bar{c}}^{b''} \mu_C(x) / x = C,
\end{aligned}$$

где

$$\bar{c} = \bar{a} + \bar{b}, \quad a'' = a + a', \quad b'' = b + b' .$$

μ_C определяется в виде $\mu_C = k_1 x + k_2$. Исходя из нормировки можно записать для $a'' \leq x \leq \bar{c}$ [4]

$$A + B = \int_{a''}^{\bar{c}} \frac{x - a''}{\bar{c} - a''} / x + \int_{\bar{c}}^{b''} \frac{b'' - x}{b'' - \bar{c}} / x = C .$$

Для остальных арифметических операций, аналогичным образом, можно получить [4]

$$A - B = \int_{a''}^{\bar{c}} \frac{x - a''}{\bar{c} - a''} / x + \int_{\bar{c}}^{b''} \frac{b'' - x}{b'' - \bar{c}} / x = C ,$$

где

$$a'' = a - b', \quad b'' = b - a', \quad \bar{c} = \bar{a} - \bar{b} .$$

Принимая функцию принадлежности в виде $\mu_c = k_1\sqrt{x} + k_2$ получим

$$A * B = \int_{a''}^{\bar{c}} \frac{\sqrt{x} - \sqrt{a''}}{\sqrt{\bar{c}} - \sqrt{a''}} / x + \int_{\bar{c}}^{b''} \frac{\sqrt{b''} - \sqrt{x}}{\sqrt{b''} - \sqrt{\bar{c}}} / x = C,$$

где

$$a'' = a * a', \quad b'' = b * b', \quad \bar{c} = \bar{a} * \bar{b}.$$

Принимая функцию принадлежности μ_C в виде $\mu_C = \frac{k_1}{x} + k_2$ получим

$$A : B = \int_{a''}^{\bar{c}} \frac{(x - a'')\bar{c}}{(\bar{c} - a'')x} / x + \int_{\bar{c}}^{b''} \frac{(b'' - x)\bar{c}}{(b'' - \bar{c})x} / x = C,$$

где

$$a'' = a' : a, \quad b'' = b' : b, \quad \bar{c} = \bar{b} : \bar{a}.$$

Примеры. А=приблизительно 6= $\tilde{6}$; В=приблизительно 8= $\tilde{8}$.

$$\tilde{6} = \int_5^6 (x-5)/x + \int_6^7 (7-x)/x,$$

при $x=5$: $\tilde{6}|_{x=5} = \int_5^6 (x-5)_{x=5} = 5-5=0;$

при $x=5,5$: $\tilde{6}|_{x=5,5} = \int_5^6 (x-5)_{x=5,5} = 5,5-5=0,5;$

при $x=6$: $\tilde{6}|_{x=6} = \int_5^6 (x-5)_{x=6} = 6-5=1;$

при $x=6,5$: $\tilde{6}|_{x=6,5} = \int_6^7 (7-x)_{x=6,5} = 7-6,5=0,5;$

при $x=7$: $\tilde{6}|_{x=7} = \int_6^7 (7-x)_{x=7} = 7-7=0.$

Следовательно

$$\tilde{6} = \{0/5; 0,5/5,5; 1/6; 0,5/6,5; 0/7\}.$$

Для $\tilde{8}$ при $x=6, 7, 8, 9$ и 10 , аналогичным образом можно получить

$$\tilde{8} = \{0/6; 0,5/7; 1/8; 0,5/9; 0/10\}.$$

Графики нечетких чисел представлены на рис.7.1. Верхние и нижние границы и вершины этих чисел следующие: для $\tilde{6}$: $a=5$, $b=7$, $\bar{a}=6$; для $\tilde{8}$: $a'=6$, $b'=10$, $\bar{b}=8$. Ниже приводятся все четыре над $\tilde{6}$ и $\tilde{8}$.

Сложение. Сначала вычислим границы и вершину суммы ($\tilde{6} + \tilde{8}$):

$$a'' = a + a' = 5 + 6 = 11; \quad b'' = b + b' = 7 + 10 = 17; \quad \bar{c} = \bar{a} + \bar{b} = 6 + 8 = 14.$$

Можно записать

$$\tilde{6} + \tilde{8} = \int_{11}^{14} \frac{x-11}{3} / x + \int_{14}^{17} \frac{17-x}{3} / x = 1\tilde{4}.$$

Вычисляя $1\tilde{4}$ при различных x : $x=12,5$; $x=15,5$ можно получить $1\tilde{4} = \{0/11; 0,5/12,5; 1/14; 0,5/15; 0/17\}$, которое графически представлено на рис.7.1.

Вычитание. Границы и вершины разности $\tilde{8} - \tilde{6}$ определяются как:

$$a'' = a' - a = 6 - 5 = 1; \quad b'' = b' - b = 10 - 7 = 3; \quad \bar{c} = \bar{b} - \bar{a} = 8 - 6 = 2.$$

Можно получить

$$\tilde{8} - \tilde{6} = \int_1^2 \frac{x-1}{3-2} / x + \int_2^3 \frac{3-x}{3-2} / x = \tilde{2}.$$

Значения функции принадлежности в точках $x=1,5$ и $x=2,5$ будут, соответственно 0,5 и 0,5. Тогда

$$\tilde{2} = \{0/1; 0,5/1,5; 1/2; 0,5/2,5; 0/3\},$$

которое графически представлено на рис.1.22.

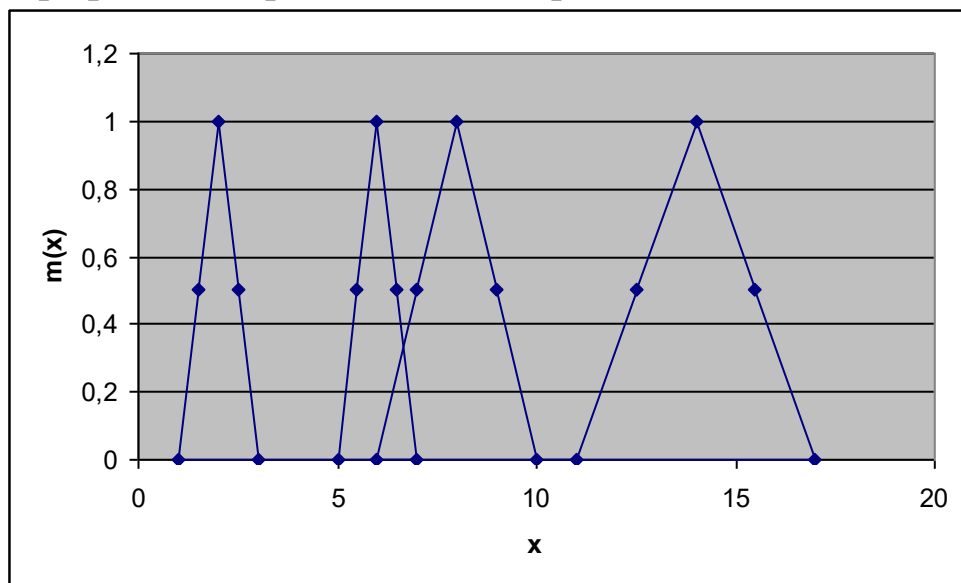


Рис.1.22. Нечеткие числа:

Приблизительно 2. Приблизительно 6.
 Приблизительно 8. Приблизительно 14.

Умножение. Определим границы и вершину $\tilde{b} \times \tilde{8}$: $a'' = a \times a' = 5 \times 6 = 30$; $b'' = b \times b' = 7 \times 10 = 70$; $\bar{c} = \bar{a} \times \bar{b} = 6 \times 8 = 48$.

Функцию принадлежности $\mu_{\tilde{b} \times \tilde{8}}$ можно представить как

$$\begin{aligned} \tilde{b} \times \tilde{8} &= \int_{30}^{48} \frac{\sqrt{x} - \sqrt{30}}{\sqrt{48} - \sqrt{30}} / x + \int_{48}^{70} \frac{\sqrt{70} - \sqrt{x}}{\sqrt{70} - \sqrt{48}} / x = \\ &= \int_{30}^{48} \frac{\sqrt{x} - 5,48}{1,45} / x + \int_{48}^{70} \frac{8,37 - \sqrt{x}}{1,44} / x = 4\tilde{8} \end{aligned}$$

Значения функции принадлежности в точках: $x=34$; $x=39$; $x=53$; $x=58$ будут, соответственно 0,24; 0,53; 0,76; 0,52. Таким образом

$\tilde{b} \times \tilde{8} = 4\tilde{8} = \{0/30; 0,24/34; 0,53/39; 0,8/44; 1/48; 0,76/53; 0,52/58; 0,26/64; 0/70\}$, которое графически представлено на рис.1.23.

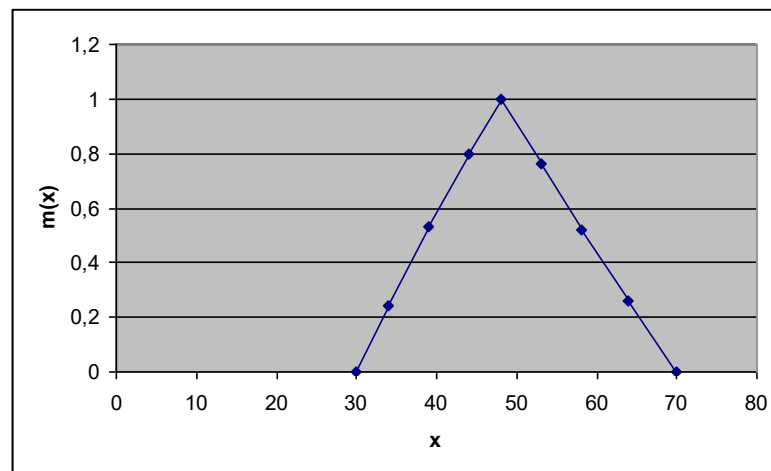


Рис.1.23. Нечеткое число
 Приблизительно 48.

Деление. Границы и вершина результата деления $\tilde{8}$ на \tilde{b} будут $a'' = a : b' = 6 : 7 = 0,86$; $b'' = b : a' = 10 : 5 = 2,0$; $\bar{c} = \bar{b} : \bar{a} = 8 : 6 = 1,33$.

Функция принадлежности определяется как

$$\begin{aligned} \tilde{8} : \tilde{b} &= \int_{0,86}^{1,33} \frac{(x - 0,86) \cdot 1,33}{(1,33 - 0,86) \cdot x} / x + \int_{1,33}^{2,0} \frac{(2,0 - x) \cdot 1,33}{(2,0 - 1,33) \cdot x} / x = \\ &= \int_{0,86}^{1,33} \frac{(x - 0,86) \cdot 1,33}{0,47 \cdot x} / x + \int_{1,33}^{2,0} \frac{(2,0 - x) \cdot 1,33}{0,67 \cdot x} / x = 1,3\tilde{3}. \end{aligned}$$

Вычисляя значения функции принадлежности в точках: $x=1,0$; $x=1,14$; $x=1,55$; $x=1,77$ и получив, соответственно, $0,39$; $0,70$; $0,60$; $0,26$ можно записать

$$\tilde{8} : \tilde{b} = \{0/0,86; 0,39/1,0; 0,70/1,14; 1/1,33; 0,60/1,55; 0,26/1,77; 0/2,0\}.$$

Графически оно представлено на рис.7.3.

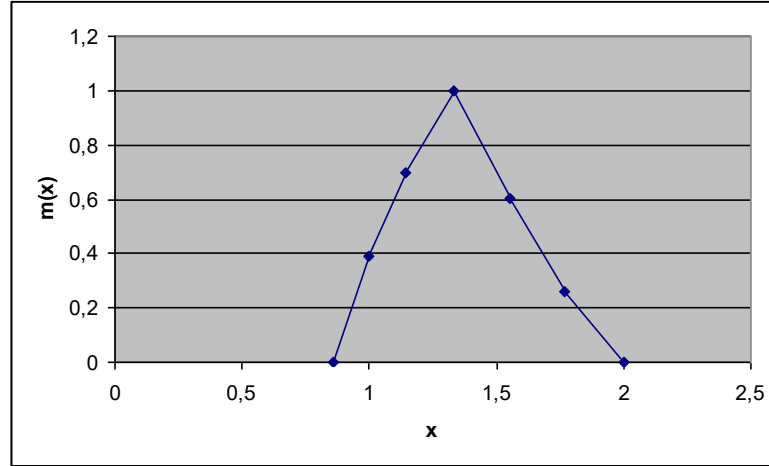


Рис.1.24. Нечеткое число
Приблизительно 1,33

Ниже рассматривается другой метод выполнения операций над нечеткими числами, основанный на использовании уровневых множеств, отличающийся значительным упрощением вычислений по сравнению с операциями на основе принципа обобщения [1-2]. При этом дополнительно следует использовать следующее определение [42]: бинарная операция $*$ на \mathbb{R} называется возрастающей, если $(x_1 > y_1, x_2 > y_2) \Rightarrow x_1 * x_2 > y_1 * y_2$. Операция $*$ называется убывающей, если $(x_1 > y_1, x_2 > y_2) \Rightarrow x_1 * x_2 < y_1 * y_2$.

Если заданы нечеткие числа A и B с функциями принадлежности μ_A и μ_B , то результат, обобщенной операции $*$ над ними есть нечеткое число $C = A * B$, задаваемое функцией принадлежности

$$\mu_C(z) = \sup_{Z=X*Y} \min(\mu_A(x), \mu_B(y)).$$

Более конкретно все четыре арифметические операции можно представить как:

Сложение.

$$\mu_{A+B}(z) = \sup_{Z=X+Y} \min(\mu_A(x), \mu_B(y)) = \sup_X \min(\mu_A(x), \mu_B(z-x)).$$

Вычитание.

$$\mu_{A-B}(z) = \sup_{Z=X-Y} \min(\mu_A(x), \mu_B(y)) = \sup_X \min(\mu_A(x), \mu_B(x-z)).$$

Умножение.

$$\mu_{A \times B}(z) = \sup_{Z=X \times Y} \min(\mu_A(x), \mu_B(y)) = \sup_X \min(\mu_A(x), \mu_B(z : x)), x \neq 0.$$

Деление.

$$\begin{aligned} \mu_{A:B}(z) &= \sup_{Z=X:Y} \min(\mu_A(x), \mu_B(y)) = \sup_X \min(\mu_A(x), \mu_B(x : z)) = \\ &= \sup_Y \min(\mu_A(yz), \mu_B(y)). \end{aligned}$$

Если нечеткие числа А и В можно представить в виде

$$A = \{\omega_1 / x_{11}; \omega_2 / x_{21}; \omega_1 / x_{12}\}; \quad B = \{\omega_1 / y_{11}; \omega_2 / y_{21}; \omega_1 / y_{12}\},$$

то результатом обобщенной операции * над ними будет нечеткое число

$$C = A * B = \{\omega_1 / (x_{11} * y_{11}); \omega_2 / (x_{21} * y_{21}); \omega_1 / (x_{12} * y_{12})\}.$$

Это справедливо, когда операция * является возрастающей либо убывающей. Операции вычитания и деления не являются такими, однако их можно представить следующим образом [2]:

$$A - B = A + (-B); \quad A : B = A \times (B^{-1}).$$

Примеры. Заданы два нечетких числа

$$\tilde{2} = \{0/1; 0,5/1,5; 1/2; 0,5/2,5; 0/3\},$$

$$\tilde{3} = \{0/2; 0,5/2,5; 1/3; 0,5/3,5; 0/4\}.$$

Ниже над ними будут выполнены четыре обобщенных арифметических операций (+, -, ×, :).

Сложение.

$$\begin{aligned} \tilde{3} + \tilde{2} &= \{0/(2+1); 0,5/(2,5+1,5); 1/(3+2); 0,5/(3,5+2,5); \\ &0/(4+3)\} = \{0/3; 0,5/4; 1/5; 0,5/6; 0/7\}. \end{aligned}$$

Произведение.

$$\tilde{3} \times \tilde{2} = \{0/2; 0,5/3,75; 1/6; 0,5/8,75; 0/12\}.$$

Вычитание.

 Определим

$$-\tilde{2} = \{0/(-3); 0,5/(-2,5); 1/(-2); 0,5/(-1,5); 0/(-1)\};$$

$$\tilde{3} - \tilde{2} = \tilde{3} + (-\tilde{2}) = \{0/(2-3); 0,5/(2,5-2,5); 1/(3-2); 0,5/(3,5-1,5);$$

$$0/(4-1)\} = \{0/(-1); 0,5/0; 1/1; 0,5/2; 0/3\}.$$

Деление.

 Определим.

$$\tilde{2}^{-1} = \{0/(1:1); 0,5/(1:1,5); 1/(1:2); 0,5/(1:2,5); 0/(1:3)\} =$$

$$= \{0/1; 0,5/0,66; 1/0,5; 0,5/0,4; 0/0,33\} = \{0/0,33; 0,5/0,4; 1/0,5; 0,5/0,66; 0/1\};$$

$$\tilde{3} : \tilde{2} = 3 \times (2^{-1}) = \{0/0,66; 0,5/1; 1/1,5; 0,5/2,33; 0/4\}.$$

Операции дополнительного вычитания и деления. В решении нечетких уравнений пригодится вычислять противоположные и обратные нечеткие числа [2]. Рассмотренные выше арифметические

операции, основанные на принципе обобщения, не позволяют отыскать противоположное число A' (такое, что $A+A'=0$) и обратное число A'' ($A \times A''=1$). Так же имеют место неравенства

$$(A-B)+B \neq A; (A:B) \times B \neq A.$$

Для точного решения уравнения

$$AX+B=D,$$

где A, B, D – нечеткие числа, X – неизвестное, используется операции дополнительного вычитания ($--$) и дополнительного деления ($//$).

В частном случае решение будет

$$X=D--B.$$

Носителями множества B и D являются , соответственно, интервалы $S_B=[b_1,b_2]$ и $S_D=[d_1,d_2]$. А носитель множества X , определяемого дополнительным вычитанием, будет

$$S_X=[d_1-b_1,d_2-b_2],$$

а через функции принадлежности как [4]

$$\mu_X(x) = \inf_z \begin{cases} 1, & \text{если } \mu_D(z-x) < \mu_B(z); \\ \mu_D, & \text{если } \mu_B(z-x) \geq \mu_D(z). \end{cases}$$

Рассматриваемая операция вычитания определена тогда, когда длина интервала носителя уменьшаемого больше, чем у вычитаемого.

Дополнительное деление. Решением уравнения $AX=B$, будет множество $X=B//A$. Если носителями множеств A и B являются

$S_A=[a_1,a_2]$ и $S_B=[d_1,d_2]$, то носитель множества X определится через функции принадлежности

$$S_X=[d_1,d_2]//[a_1,a_2] = \begin{cases} d_1 : a_1, d_2 : a_2, & \text{если } S_A > 0; S_D > 0 \\ d_1 : a_2, d_2 : a_1, & \text{если } S_A > 0; S_D < 0 \\ d_2 : a_1, d_1 : a_2, & \text{если } S_A < 0; S_D > 0 \\ d_2 : a_2, d_1 : a_1, & \text{если } S_A < 0; S_D < 0 \end{cases}$$

$$\mu_X(x) = \inf_t \begin{cases} 1, & \text{если } \mu_A(t/x) < \mu_B(t), \\ \mu_B(t), & \text{если } \mu_A(t/x) \geq \mu_B(t). \end{cases}$$

Эта операция определена не для любых нечетких чисел A и B , а для таких, у которых интервалы носители удовлетворяют определенным условиям [2].

Примеры. Решить уравнение

$$X+B=D,$$

при $B = \tilde{8} = \{0/6; 0,5/7; 1/8; 0,5/9; 0/10\}$,

$D = 1\tilde{4} = \{0/10; 0,5/12; 1/14; 0,5/16; 0/18\}$,

функции принадлежности которых представлены на рис.1.25.

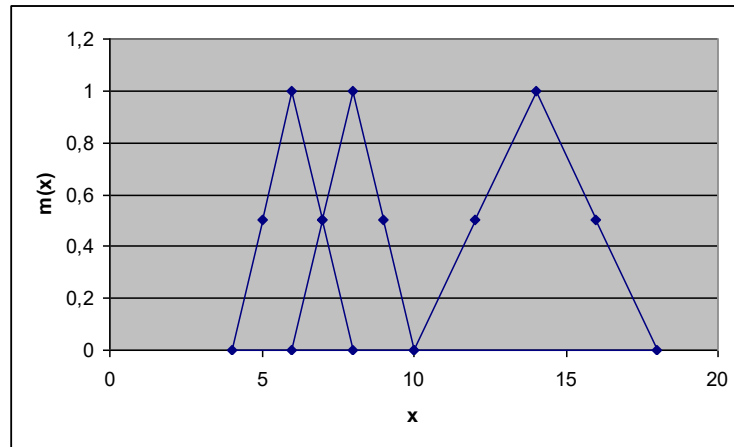


Рис.1.25. функции принадлежности множеств для операции дополнительного вычитания.

Интервалы-носители для B и D будут $S_B = [6,10]$; $S_D = [10,18]$.
 $S_X = [4,8]$. Можно определить функцию принадлежности $\mu_X(x)$ графически представленную на рис. 1.25.

Соответственно, $X = (0/4; 0,5/5; 1,0/6; 0,5/7; 0/8)$.

При $A = \tilde{8} = \{0/6; 0,5/7; 1/8; 0,5/9; 0/10\}$ и $D = 2\tilde{4} = \{0/6; 0,5/14; 1,0/24; 0,5/36; 0/50\}$ решить уравнение

$$AX=D.$$

Функции принадлежности μ_A и μ_D представлены на рис.1.26.

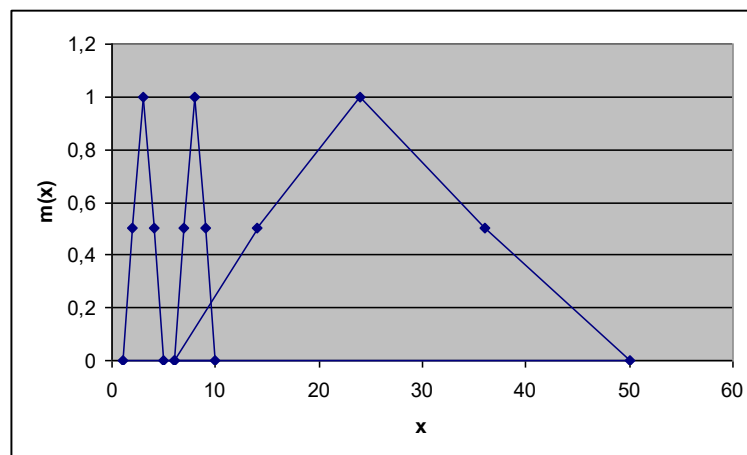


Рис.1.26. Функции принадлежности множеств для операции дополнительного деления

Интервалы-носители множеств A и D будут $S_A = [6,10]$; $S_D = [6,50]$. $S_X = [6 : 6,50 : 10] = [1,5]$.

Можно определить значение функции принадлежности $\mu_X(x)$, которая представлена на рис.1.26.

Решением уравнения является

$$X = \{0/1; 0,5/2; 1/3; 0,5/4; 0/5\}.$$

Нечеткие числа L-R типа

Функция, обычно обозначаемая через L или R, является функцией, представляющей нечеткое число, если:

- 1) $L(x) = L(-x)$;
- 2) $L(0) = 1$;
- 3) L является невозрастающей на интервале $[0, \infty)$.

Примеры.

а) $L(x) = \begin{cases} 1, & x \in [-1, 1]; \\ 0, & \text{в противном случае.} \end{cases}$

б) $L(x) = \max(0; 1 - |x|^p)$, $p \geq 0$.

в) $L(x) = \theta^{-|x|^p}$, $p > 0$.

г) $L(x) = 1/(1 + |x|^p)$, $p \geq 0$.

Нечеткое число M называется L-R типа, если:

$$\mu_M(x) = \begin{cases} L\left(\frac{m-x}{\alpha}\right) & \text{для } x \leq m, \alpha > 0; \\ R\left(\frac{x-m}{\beta}\right) & \text{для } x \geq m, \beta > 0. \end{cases}$$

L является левым представлением; R – правым представлением; m – средним значением M; α и β называются соответственно, левым и правым растяжением. Когда растяжения равны нулю, M является обычным числом. По мере увеличения растяжения, M становится все более нечетким. Символическая запись числа L-R типа $M = (m, \alpha, \beta)_{LR}$.

Рассмотрим алгебраические действия с нечеткими числами типа L-R.

Теорема 1.1. [1] Пусть M и N – два нечетких числа с непрерывными функциями принадлежности; * - непрерывно возрастающая бинарная операция; $[\lambda_M, \rho_M]$ - некоторый интервал (возможно, что $\lambda_m = \rho_m$) на котором функция принадлежности нечеткого числа M не убывает; $[\lambda_N, \rho_N]$ - аналогичный интервал для N, причем

$$\mu_M(x) = \mu_N(y) = \omega \text{ для } \forall x \in [\lambda_M, \rho_M], \forall y \in [\lambda_N, \rho_N].$$

Когда нечеткое множество $M * N$, функция принадлежности которого определяется согласно принципу обобщения, является нечетким числом с непрерывной функцией принадлежности, функция принадлежности этого нечеткого числа в любой точке $t \in [\lambda_M * \lambda_N, \rho_M * \rho_N]$ определяется по формуле $\mu_{M*N}(t) = \omega$.

Сложение нечетких чисел. Рассмотрим возрастающие части двух нечетких чисел

$$M = (m, \alpha, \beta)_{LR} \text{ и } N = (n, \gamma, \delta)_{LR}.$$

Пусть x и y – единственные вещественные числа, такие, что

$$L\left(\frac{m-x}{\alpha}\right) = \omega = L\left(\frac{n-y}{\gamma}\right),$$

где ω - фиксированная величина на отрезке $[0,1]$. Это эквивалентно тому, что

$$x = m - \alpha L^{-1}(\omega), \quad y = n - \gamma L^{-1}(\omega), \quad (1.31)$$

Из (1.31) следует, что

$$z = x + y = m + n - (\alpha + \gamma)L^{-1}(\omega) \text{ и } L\left(\frac{m+n-z}{\alpha+\gamma}\right) = \omega.$$

Аналогично, для убывающих частей M и N получим

$$R\left(\frac{z-(m+n)}{\beta+\delta}\right) = \omega.$$

Отсюда, с учетом теоремы, следует, что

$$(m, \alpha, \beta)_{LR} \oplus (n, \gamma, \delta)_{LR} = (m+n, \alpha+\gamma, \beta+\delta)_{LR}. \quad (1.32)$$

В более общем случае

$$(m, \alpha, \beta)_{LR} \oplus (n, \gamma, \delta)_{LR} = (m+n, 1, 1)_{L''R''},$$

где $L'' = (\alpha L^{-1} + \gamma L^{-1})^{-1}$, $R'' = (\beta L^{-1} + \delta L^{-1})^{-1}$.

Формула для отрицания нечеткого числа имеет вид

$$-(m, \alpha, \beta)_{LR} = (-m, \beta, \alpha)_{RL}. \quad (1.33)$$

Заметим, что функции L и R , представляющие нечеткое число, поменялись местами. Из (1.32) и (1.33) можно вывести формулу для вычитания

$$(m, \alpha, \beta)_{LR} - (n, \gamma, \delta)_{RL} = (m-n, \alpha+\delta, \beta+\gamma)_{LR}.$$

Умножение. Аналогично предыдущему можно получить

$$z = xy = mn + (m\gamma + n\alpha)L^{-1}(\omega) + \alpha\gamma(L^{-1}(\omega))^2. \quad (1.34)$$

Уравнение (1.34) является уравнением второго порядка относительно $L^{-1}(\omega)$ и имеет один положительный корень в случае $z \leq mn$. Используя теорему, можно определить функцию

принадлежности произведения $\mu_{M \otimes N}$. Обычно это произведение $M \otimes N$ не бывает нечетким числом типа L-R. Тем не менее, если α и β являются малыми по сравнению с m и n и (или) ω находится в окрестности 1, то можно пренебречь членом $\alpha\gamma(L^{-1}(\omega))^2$. При этом уравнение упрощается и можно получить приближенные формулы

$$(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, m\gamma + n\alpha, m\delta + n\beta)_{LR}, \quad \text{для } M > 0, N > 0,$$

$$(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, n\alpha - m\delta, n\beta - m\gamma)_{LR}, \quad \text{для } M < 0, N > 0,$$

$$(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} = (mn, -n\beta - m\delta, n\alpha - m\gamma)_{LR}, \quad \text{для } M < 0, N < 0.$$

Если растяжения не малы, по сравнению со средними значениями, можно использовать другие приближенные формулы. Например, если $M > 0$ и $N > 0$ то

$$(m, \alpha, \beta)_{LR} \otimes (n, \gamma, \delta)_{LR} \approx (mn, m\gamma + n\alpha - \alpha\gamma, m\delta + n\beta + \beta\delta)_{LR}.$$

Функция принадлежности, определяемая этим равенством, совпадает с μ , по крайней мере, в трех точках:

$$(mn, 1); [(m - \alpha)(n - \gamma), L(1)]; [(m + \beta)(n + \delta), R(1)].$$

Умножение на скаляр. Имеют место формулы:

$$\forall \lambda > 0 \lambda \in R: \lambda \otimes (m, \alpha, \beta)_{LR} = (\lambda m, \lambda \alpha, \lambda \beta)_{LR},$$

$$\forall \lambda < 0 \lambda \in R: \lambda \otimes (m, \alpha, \beta)_{LR} = (\lambda m, -\lambda \alpha, -\lambda \beta)_{LR}.$$

Деление. Для положительных, нечетких чисел L-R и R-L типа получены следующие приближенные результаты:

$$(m, \alpha, \beta)_{LR} \div (n, \gamma, \delta)_{LR} \approx \left(\frac{m}{\tilde{n}}, \frac{\delta m + \alpha n}{n^2}, \frac{\gamma m + \beta n}{n^2} \right)_{LR}.$$

Сравнение нечетких чисел

При сравнении нечетких чисел возникают вопросы двух типов:

Каково нечеткое значение наибольшего (наименьшего) из заданного семейства нечетких чисел?

Чему равно значение истинности высказывания « \tilde{m} больше (меньше), чем \tilde{n} ». Другими словами, какова возможность того, что \tilde{m} больше (меньше), чем \tilde{n} ?

Для ответа на первый вопрос воспользуемся принципом обобщения применительно к функциям

$$z(m, n) = \max\{m; n\};$$

$$t(m, n) = \min\{m; n\}.$$

Если $\mu_1(m), \mu_2(n)$ - функции принадлежности нечетких чисел \tilde{m} и \tilde{n} , то в соответствии с принципом обобщения получим

$$\mu_3(z) = \sup_{z=\max(m;n)} \min\{\mu_1(m), \mu_2(n)\};$$

$$\mu_4(t) = \sup_{t=\min(m;n)} \min\{\mu_1(m), \mu_2(n)\}.$$

Нечеткие множества \tilde{z} и \tilde{t} с функциями принадлежности μ_3 и μ_4 являются выпуклыми нормальными числами, вообще говоря, отличными от \tilde{m} и \tilde{n} . Постараемся ответить на второй вопрос. Степень истинности высказывания « \tilde{n} больше, чем \tilde{m} », которое будем записывать как $v(\tilde{n} > \tilde{m})$, определяется следующим образом:

$$v(\tilde{n} > \tilde{m}) = \max_{n \geq m} \min\{\mu_1\}.$$

Несложно проверить, что для случая « \tilde{n} больше, чем \tilde{m} »

$$v(\tilde{n} > \tilde{m}) = \min\{\mu_1(A), \mu_2(B)\} = \min\{1,1\} = 1;$$

$$v(\tilde{m} > \tilde{n}) = \min\{\mu_1(C), \mu_2(C)\} = d.$$

Число d характеризует степень различимости чисел \tilde{m} и \tilde{n} . Чем ближе к 1 ордината точки C , тем сложнее ответить на вопрос, какое из чисел \tilde{m} и \tilde{n} больше.

Пусть \tilde{m} и \tilde{n} являются нечеткими числами типа L-R и R-L, соответственно; $\tilde{m} = (m, \alpha, \beta)$, $\tilde{n} = (n, \gamma, \delta)$, тогда C находится из уравнения

$$(C - m) / \beta = (n - C) / \gamma,$$

откуда

$$C = (\beta n + \gamma m) / (\beta + \gamma) \text{ и } d = R[(n - m) / (\beta + \gamma)]. \quad (1.35)$$

Таким образом, высота нечеткого множества, являющегося пересечением двух нечетких чисел, определяется по формуле (1.35) (напомним, что мы рассматриваем случай $m < n$). Следовательно, для того, чтобы сравнить \tilde{m} и \tilde{n} , требуется знать как $v(\tilde{m} > \tilde{n})$, так и $v(\tilde{m} < \tilde{n})$. Если например, $v(\tilde{m} > \tilde{n}) = 1$, то это означает, что или \tilde{m} больше, чем \tilde{n} или оба нечетких числа слишком близки друг к другу, чтобы их можно было бы разделить. В этом случае можно выбрать некоторое пороговое значение τ и согласиться, что \tilde{m} больше \tilde{n} с уровнем τ , если $v(\tilde{n} > \tilde{m}) \leq \tau$. Это обозначается как $\tilde{m} \underset{\tau}{>} \tilde{n}$.

Для нечетких чисел типа L-R правило может быть определено следующим образом:

$$\tilde{n} > \tilde{m} \Leftrightarrow n - m \geq \beta + \gamma; (\tau = R(1));$$

$$\tilde{m} > \tilde{n} \Leftrightarrow m - n \geq \alpha + \delta; (\tau = L(1));$$

где для краткости обозначено $>$ вместо $\underset{\tau}{>}$. Можно ввести и другое определение для «n, больше m»:

$$\tilde{n} \geq \tilde{m} \Leftrightarrow \max(\tilde{m}, \tilde{n}) = \tilde{n} \text{ и } \min(\tilde{m}, \tilde{n}) = \tilde{m}.$$

Примеры.

1. Нечеткое число \tilde{n} больше нечеткого числа \tilde{m} , если

$$\begin{aligned} n &> m \\ n - \underline{n} &\geq m - \underline{m} \\ n + \overline{n} &\geq m + \overline{m} \end{aligned} \tag{1.36}$$

2. Нечеткое число \tilde{m} меньше нечеткого числа \tilde{n} , если

$$\begin{aligned} m &\leq n \\ m - \underline{m} &\leq n - \underline{n} \\ m + \overline{m} &\leq n + \overline{n} \end{aligned} \tag{1.37}$$

3. Нечеткое число \tilde{n} равно нечеткому числу \tilde{m} , если

$$\begin{aligned} n &= m \\ n - \underline{n} &= m - \underline{m} \\ n + \overline{n} &= m + \overline{m} \end{aligned} \tag{1.38}$$

Пример. Заданы два нечетких числа L-R типа:

- a) $A = (1,5,2)_{LR}$ $B = (3,8,4)_{LR}$
- b) $A = (8,10,6)_{LR}$ $B = (6,9,5)_{LR}$
- c) $A = (20,35,30)_{LR}$ $B = (18,20,25)_{LR}$
- d) $A = (4,7,6)_{LR}$ $B = (8,10,5)_{LR}$
- e) $A = (2,8,6)_{LR}$ $B = (4,8,6)_{LR}$

Вычислить сложение, вычитание, умножение и деление этих чисел.

Ответ:

Сложение	Вычитание	Умножение	Деление
a) $(4,13,6)_{LR}$	$(-2,9,10)_{LR}$	$(3,23,10)_{LR}$	$(0.3,2.11,1.44)_{LR}$
b) $(14,19,11)_{LR}$	$(-2,15,15)_{LR}$	$(48,132,76)_{LR}$	$(1.33,2.78,3)_{LR}$
c) $(38,55,55)_{LR}$	$(2,60,50)_{LR}$	$(360,1030,1040)_{LR}$	$(1.11,3.49,2.9)_{LR}$
d) $(12,17,11)_{LR}$	$(-4,12,16)_{LR}$	$(32,96,68)_{LR}$	$(0.5,1.19,1.38)_{LR}$
e) $(6,16,12)_{LR}$	$(-2,14,14)_{LR}$	$(8,48,36)_{LR}$	$(0.5,2.75,2.5)_{LR}$

Нечеткий анализ

Нечеткая функция

Сначала дадим принцип расширения Заде, который является основополагающей концепцией в теории нечетких множеств, предназначенной для обобщения классических математических методов для нечетких множеств [1-2].

Допустим, что X является универсумом, A – нечетким множеством в X , а f отображением из X в Y , $y=f(x)$. Тогда принцип расширения дает возможность определить нечеткое множество B в Y как

$$B = f(A) = \{(y, \mu_B(y)) \mid y = f(x), x \in X\}$$

где

$$\mu_B(y) = \begin{cases} \sup_{x \in f^{-1}(y)} \mu_A(x), & \text{если } f^{-1}(y) \neq \emptyset, \\ 0, & \text{в других случаях.} \end{cases}$$

Пример.

$$A = \{(-1, 0.3), (0, 0.6), (1, 1), (2, 0.2)\}$$

$$f(x) = x^2.$$

Применяя принцип расширения можем найти

$$B = f(A) = \{(0, 0.6), (1, 1), (4, 0.2)\}.$$

Определение 1.17. Нечеткая функция.

Пусть дана функция (четкая) $f: R \rightarrow R$. Тогда мы можем выбрать ее график [5]

$$\{(x, y) \in R^2 \mid y = f(x)\}.$$

В качестве ядра нечеткого множества F , значения принадлежности которого могут, например, монотонно убывать с ростом расстояния от этого графика. Это нечеткое множество F представляет нечеткую функцию. Для явных функций f мы можем интерпретировать F как семейство (с параметром x) нечетких чисел $Y(x)$, каждое из которых имеет $\{f(x)\}$ в качестве ее ядра и

$$\mu_{Y(x)}(y) = \mu_F(x, y)$$

в качестве функции принадлежности.

Для случая неявной функции f с графиком

$$\{(x, y) \in R^2 \mid f(x, y) = 0\}$$

мы можем использовать частотно-вероятностную интерпретацию в качестве границы (четкой) области при переходе к нечетким неявным функциям.

Существуют три основных типа нечетких функций:

обычные функции с нечеткими свойствами или удовлетворяющие нечетким ограничениям;

функции, которые просто отражают нечеткость их аргументов, сами не внося дополнительной нечеткости: образом четкого элемента здесь является четкое значение функции;

плохо-определенные функции четкого аргумента: образ какого-либо элемента расплывается под действием функции.

Пусть f – обычная функция из V в W

$$x \in V \rightarrow f(x) \in W,$$

а V и W – два универсума.

Пусть A и B – два нечетких множества на V и W , соответственно. F называют имеющей нечеткий домен A и нечеткую область B если и только если

$$\forall x \in V, \mu_B(f(x)) \geq \mu_A(x).$$

Рассмотрим нечеткое расширение четкой функции.

Пусть f – четкая функция из V в W : образ нечеткого множества X на V определяется посредством принципа расширения.

$$\mu_{f(x)}(y) = \sup_{x \in f^{-1}(y)} \mu_X(x)$$

$$\mu_{f(x)}(y) = 0, \text{ если } f^{-1}(y) = \emptyset,$$

где $f^{-1}(y)$ - множество антецедентов y .

Допустим, что X и Y универсумы и $P(Y)$ множество всех нечетких множеств в Y .

$\tilde{f}: X \rightarrow P(Y)$ - называется нечеткой функцией, если и только если

$$\mu_{\tilde{f}(x)}(y) = \mu_R(x, y), \forall (x, y) \in X \times Y.$$

Определение 1.18. Экстремум нечеткой функции.

Допустим, что $\tilde{f}(x)$ определяется на конечной ординарной (не нечеткой) области D .

Нечеткий максимум $\tilde{f}(x)$ определяется как

$$\tilde{M} = \max_{x \in D} \tilde{f}(x) = \left\{ (\sup_{x \in D} \tilde{f}(x), \mu_{\tilde{M}}(x)) \mid x \in D \right\}.$$

Интегрирование нечетких функций

Определение 1.19. Интегрирование нечеткой функции.

$\tilde{f}(x)$ - нечеткая функция из $[a,b] \subset R$, $\forall x \in [a,b]$. $\tilde{f}(x)$ нечеткое число, $f_\alpha^-(x)$ и $f_\alpha^+(x)$ - α - уровневые срезы. Интеграл $\tilde{f}(x)$ в $[a,b]$ определяется как нечеткое множество

$$\tilde{I}(a,b) = \left\{ \left(\int_a^b f_\alpha^-(x) dx + \int_a^b f_\alpha^+(x) dx \right), \alpha \right\}.$$

где $y = \{g : [a,b] \rightarrow R / g\}$

Допустим, что нечеткая функция представляется в виде LR-нечеткого числа [3].

$$\tilde{f}(x) = (f(x), s(x), t(x))_{LR}$$

для всех $x \in [a,b]$. f, s, t предполагается интегрируемыми функциями в $[a,b]$. Тогда

$$\tilde{I}(a,b) = \left(\int_a^b f(x) dx, \int_a^b s(x) dx, \int_a^b t(x) dx \right)_{LR}.$$

Пример. Нечеткая функция $\tilde{f}(x) = (f(x), s(x), t(x))_{LR}$ с $f(x) = x^2$, $s(x) = x/4$ и $t(x) = x/2$.

$$L(x) = \frac{1}{1+x^2}, \quad R(x) = \frac{1}{1+2|x|}.$$

Требуется найти $\int_1^4 \tilde{f}$. Найдем следующие интегралы

$$\int_1^4 x^2 dx = 21; \quad \int_1^4 \frac{x}{4} dx = 1/875; \quad \int_1^4 \frac{x}{2} dx = 3.75.$$

Тогда значение нечеткого интеграла $\tilde{I}(a,b)$ будет

$$\tilde{I}(a,b) = (21, 1.875, 3.75)_{LR}.$$

Рассмотрим интегрирование четкой функции на нечетком интервале, концами которого являются $\mu_a(x), \mu_b(x)$. $\mu_a(x)$ и $\mu_b(x)$ могут интерпретироваться как степени нижней и верхней границы нечеткой области \tilde{D} . Допустим, что f обычная интегрируемая функция в интервале $J = [a_0, b_0]$. Согласно принципу расширения функция принадлежности интеграла $\int_{\tilde{D}} f$ определяется как

$$\mu_{\int_{\tilde{D}} f}(Z) = \sup_{x,y \in J} \min(\mu_a(x), \mu_b(y))$$

$$Z = \int_x^y f.$$

Пример.

$$a = \{(4,0.8), (5,1), (6,0.4)\}$$

$$b = \{(6,0.7), (7,1), (8,0.2)\}$$

$$f(x) = 2, x \in [4,8].$$

Тогда

$$\int_{\tilde{D}} f(x)dx = \int_4^8 2dx = 2x \Big|_4^8$$

Детальные вычислительные результаты приведены ниже [3]

Таблица 1.3

(a,b)	$\int_a^b 2dx$	$\min(\mu_x(a), \mu_x(b))$
(4,6)	4	0.7
(4,7)	6	0.8
(4,8)	8	0.2
(5,6)	2	0.7
(5,7)	4	1.0
(5,8)	6	0.2
(6,6)	0	0.4
(6,7)	2	0.4
(6,8)	4	0.2

Выбирая максимальное значение функции принадлежности для каждого значения интеграла получим

$$\int_{\tilde{D}} f \{(0,0.4), (2,0.7), (4,1), (6,0.8), (8,0.2)\}.$$

Дифференцирование нечетких функций

Определение 1.20. Нечеткое дифференцирование.

Функция принадлежности нечеткого множества «производная функции в нечеткой точке \tilde{X}_0 » согласно принципа расширения определяется как

$$\mu(y) = \sup_{x \in f^{-1}(y)} \mu_{\tilde{X}_0}(x)$$

Пример. $f(x) = \frac{3}{5}x^5$

$$\tilde{X}_0 = \{(-1,0.3), (0,1), (1,5)\}$$

$f'(x) = 3x^4$. Искомая производная действительной функции $f(x)$ в точке \tilde{x}_0 будет как

$$f'(x_0) = \{(0,1), (3,0.6)\}.$$

Отметим следующие свойства нечеткой производной [3].

Если f' и g' непрерывные и обе убывающие или возрастающие, то

$$f'(\tilde{x}_0) \oplus g'(\tilde{x}_0) = (f' + g')(\tilde{x}_0)$$

$$(f \cdot g)'(\tilde{x}_0) = (f'g + fg')(\tilde{x}_0) \subseteq [f'(\tilde{x}_0) \otimes g(\tilde{x}_0)] \oplus [f(\tilde{x}_0) \cdot g'(\tilde{x}_0)].$$

Если f, g, f' и g' непрерывные, f и g положительные и f' и g' являются неубывающими (f, g – отрицательные и f', g' – не возрастающие), то

$$(f \cdot g)'(\tilde{x}_0) = [f'(\tilde{x}_0) \otimes g(\tilde{x}_0)] \oplus [f(\tilde{x}_0) \cdot g'(\tilde{x}_0)]$$

Производная нечеткой функции F из R_0 в точке x определяется как

$$\mu_{F(x_0)}(y) = \sup\{\alpha \mid y = (f_\alpha^q)'(x_0), q \in \{+, -\}\}$$

где предполагается существование, ограниченность и дифференцируемость f_α^+, f_α^- для всех $\alpha \in [0,1]$.

Нечеткие уравнения

Определение 1.21. Нечеткие уравнения.

В общем случае, нечеткими уравнениями называются уравнения, в которых коэффициенты и/или переменные являются нечеткими числами.

В практике часто встречаются уравнения с обычными математическими термами и нечеткими математическими отношениями и уравнения с нечеткими числами и обычными математическими отношениями.

Если f_1 и f_2 математические термы (конструкции из элементов $x \in R^1$ и связывающих операций: $+, \times, -, :$), Q нечеткое отношение, то

$$f_1 Q f_2$$

называется нечетким уравнением с нечетким отношением.

Примером Q может являться Q_Δ «приблизительно равно».

Если f_1 и f_2 являются нечеткими термами (конструкции из элементов $\mu_{A_i} \in F(R^1)$, $i \in N$, связанных с операциями: $\oplus, \approx, \otimes, \text{m}\tilde{\text{a}}\text{x}, \text{m}\tilde{\text{i}}\text{n}$),

а R обычным математическим отношением, то используя α -срезы можно определить следующее уравнение

$$\left(\bigcup_{\alpha} \alpha f_1 \right) R \left(\bigcup_{\alpha} \alpha f_2 \right) = \left(\bigcup_{\alpha} \alpha [\delta_{f_1}, \gamma_{f_1}] \right) R \left(\bigcup_{\alpha} \alpha [\delta_{f_2}, \gamma_{f_2}] \right),$$

где $\delta_{f_1} \triangleq \delta_{f_1}(\alpha)$; $\delta_{f_2} \triangleq \delta_{f_2}(\alpha)$; $\gamma_{f_1} \triangleq \gamma_{f_1}(\alpha)$; $\gamma_{f_2} \triangleq \gamma_{f_2}(\alpha)$.

1.3. Предварительная обработка и оценка качества изображений на основе аппарата нечетких множеств

В работе рассмотрены вопросы, связанные с предварительной обработкой изображений на основе аппарата нечетких множеств. Предложена архитектура системы обработки изображений, а также приведены основные этапы обработки цифровых изображений с использованием аппарата нечетких множеств и материалы по формированию базы правил. Рассмотрены вопросы улучшения яркости и контраста изображений с использованием аппарата нечетких множеств. Предложен алгоритм количественной оценки качества изображения методами нечеткой логики.

Известно, что системы обработки данных (СОД), разрабатываемые в последние годы, характеризуются высоким уровнем использования различных алгоритмов и технологий интеллектуализации процессов обработки данных. Одним из основных направлений в области интеллектуализации СОД является системы обработки информации, представленных в виде цифровых изображений.

Технологии интеллектуализации процессов обработки изображений востребованы при создании систем видеонаблюдения и контроля доступа. В основе подобных систем лежат методы и алгоритмы, связанные с вопросами биометрической идентификации личности [11-15]. В прикладных системах анализа и обработки изображений основными функциональными задачами являются оценки качества изображения, определения границ объекта, классификация, кластеризация и распознавание образов.

Основные методологические концепции в вопросах обработки изображений долгое время основывались на различных модификациях спектральных методов, а трудности при решении прикладных задач, относились на счет недостатков оборудования и аппаратного обеспечения – нехватки памяти и низкого

быстродействия. В настоящее время научно-технический прогресс привел к резкому увеличению возможностей вычислительной техники и построению на ее основе специализированных устройств, что, казалось бы, должно было привести к решению основных задач восприятия изображений техническими системами. Однако проблема не исчезла, а лишь возросло понимание ее значимости и специфических трудностей. Сложившаяся ситуация вынуждает исследователей делать все больший упор на развитие новых подходов и способов в обработке изображений, чтобы найти пути преодоления разрыва между «машинным» и «человеческим» восприятием.

В последние годы более интенсивно развивается применение нечетких методов в задачах анализа и обработки изображений [16-23], что связано со следующими: 1) эти методы являются мощными инструментами для представления и обработки знаний; 2) они могут управлять неопределенностью и неоднозначностью эффективно. Во многих приложениях обработки изображений требуется использовать экспертные знания, чтобы преодолеть некоторые трудности (например, распознавание объектов, анализ сцены). В теории нечетких множеств и нечеткой логики имеются мощные инструменты для представления и процесса человеческого познания в виде нечетких ЕСЛИ-ТО правил [24]. Нечеткие правила конструируются из функции принадлежности входов и выходов [24 - 26]. Входы и выходы, приведенные к нечеткому виду, облегчают выделение правил и их обобщение. В общем случае правила могут быть получены из опыта одного или нескольких экспертов в рассматриваемой предметной области. Несомненно, что определение хороших лингвистических правил зависит от объема и качества знаний эксперта. Однако в настоящее время не существует формализованного подхода к задаче точного и однозначного представления знаний в рамках теории нечетких множеств [27-29]. В связи с этим, представление знаний в фаззифицированном виде, по существу, произвольно.

Целью данной работы является разработка алгоритмов подавления шума усредняющими фильтрами и оценка качества изображения.

Для достижения цели необходимо решить следующие задачи:

- 1) проанализировать состояние вопросов улучшения изображений и сформулировать задачи исследования;
- 2) разработать алгоритмы улучшения и оценки качества изображения на основе теории нечетких множеств;
- 3) провести экспериментальные исследования для проверки работоспособности разработанных алгоритмов.

Качество полутонового изображения зависит от многих параметров, основными из которых являются [30-33]:

- яркость пикселя изображения представляет собой характеристику, определяющую его меры различия от чёрного цвета;
- контрастность можно количественно оценить как разность максимального и минимального значений яркости изображения. Другими словами, она представляет собой характеристику того, насколько большой разброс имеют цвета пикселей изображения;
- шум, которого можно количественно оценить как средний арифметический разброс значений яркости на изображении.

Основными недостатками изображений, в большинстве случаев, являются искаженные яркостные характеристики, шум и низкая контрастность.

Поэтому основная цель методов улучшения состоит в преобразовании изображений к такому виду, что делает их более контрастными и, соответственно, более информативнее [34-36].

В последние годы в развитых странах мира ведутся исследования по использованию нечетких методов в обработке изображений, что связано со следующими: 1) эти методы являются мощными инструментами для представления и обработки знаний; 2) они могут управлять неопределенностью и неоднозначностью эффективно. Во многих приложениях обработки изображений требуется использовать экспертные знания, чтобы преодолеть некоторые трудности (например, распознавание объектов, анализ сцены) [37, 38]. В теории нечетких множеств и нечеткой логики имеются мощные инструменты для представления и процесса человеческого познания в виде нечетких ЕСЛИ-ТО правил. С другой стороны, многие трудности в обработке изображений возникают из-за случайности, неоднозначности и неопределенности в данных, которые используется в рассматриваемых задачах. Для работы со случайностями в

обработке изображений могут быть использована теория вероятности, для других видов несовершенства, например, геометрическую нечеткость можно представить на основе аппарата нечетких множеств и нечеткой логики [36-41].

В [36] один из первых, где рассмотрены вопросы повышения качества с помощью аппарата нечеткой логики. В ней описан метод выделения нечетких свойств изображения в градациях серого, которого можно применить для повышения контрастности этого изображения. Повышение качества исходного изображения обычно является одним из первых этапов в задачах компьютерного зрения. Методы повышения качества изображения, как правило, позволяют удалить шум, сгладить регионы, где уровни серого существенно не меняются, и подчеркнуть резкие изменения уровней серого.

Так как аппарат нечеткой логики позволяет включить в себя эвристические знания о его конкретном применении в виде правил, он идеально подходит для построения системы улучшения изображения. Это привело к разработке различных методов улучшения изображения на основе нечеткой логики. Далее кратко рассмотрим некоторые из них.

В [37] предлагается фильтр для динамического уменьшения сужения диапазона значений яркости и повышения контраста, используя подход, основанный на нечетких правилах. Метод основан на алгоритме, приведенном в [38]. В [39] предложен нелинейный нечеткий фильтр для обработки изображений. Известно, что усредняющие фильтры эффективно удаляют гауссов шум, а фильтры, основанные на порядковых статистиках, такие как медианный фильтр, эффективно используются для удаления импульсного шума. Для объединения этих двух фильтров в [40] использована нечеткая логика.

Несмотря на достигнутые успехи в области цифровой обработки изображений с использованием аппарата нечетких множеств, имеется ряд нерешенных задач. К их числу можно отнести проблему адекватного отображения предметной области на нечеткую систему, выбора моделей нечеткого логического вывода и их интеграции в единую интеллектуальную систему [38-43].

Рассмотрим изображение F размера $M \times N$ пиксель, наличие серого уровня r в диапазоне между 0 и $L-1$. При применении нечеткого множества для обработки изображений они

рассматриваются как массив нечетких синглтонов. Каждый элемент массива указывает на значение принадлежности $\mu_F(f(x, y))$ серого уровня $f(x, y)$, соответствующее (x, y) -му пикселю, в соответствии с predetermined свойствами изображения, такими как яркость, резкость, однородность [36-38].

В виде обобщения этого подхода введем следующее представление изображения в нечеткой среде.

Изображение F , описанное в нечеткой среде, имеет вид

$$F = \{ \langle f(x, y), \mu_F(f(x, y)) \rangle \mid f(x, y) \in \{0, \dots, L-1\} \},$$

где $x \in \{1, \dots, M\}$, $y \in \{1, \dots, N\}$, $\mu_F(f(x, y))$ обозначают соответственно степень принадлежности (x, y) -го пикселя к множеству в соответствии со свойствами изображения.

База знаний состоит из определённого набора нечетких правил. Обычно число правил, требуемых в системе, связано с числом управляемых параметров. Например, нечеткие правила для сглаживания резкости и выделения контуров на изображении в градациях серого, которые состоят из трех входов и столько же выходов [44, 45]:

ЕСЛИ пиксель темнее соседних пикселей

ТО сделать его светлее

ИНАЧЕ

ЕСЛИ пиксель светлее соседних пикселей

ТО сделать его темнее

ИНАЧЕ оставить без изменения.

В этих правилах различия значений яркостей между рассматриваемым пикселем и его соседями являются входными данными, а увеличение/уменьшение значений яркостей этого пикселя является выходным данным. Если предположить, что значения яркости изображения находятся в интервале $[0, L-1]$ (где L - число градаций яркости), то простые треугольные нечеткие множества – среднее положительное и среднее отрицательное определены на интервале $[-L+1, L-1]$ и представляют яркие и темные значения яркостей входных переменных, а малый положительный, нулевой и небольшой отрицательный треугольные нечеткие множества определены как приращение указанного значения по следствиям составленных правил. Выходное значение добавляется к исходному значению яркости пикселя.

Правила или нечеткие ассоциации представляют собой знания, которые важны для системы с точки зрения отклика на всевозможные комбинации входов. Нечеткие системы запоминают наборы нечетких ассоциаций или правил [44, 45].

Для количественной оценки указанных параметров в качестве основных входных переменных выступают факторы, которые оказывают влияние на качество изображения. Само качество обрабатываемого изображения, является искомой выходной переменной.

Пусть нечеткая переменная описывается набором F — универсальное множество (область рассуждений), A — нечеткое множество на F . База правил, содержащая нечеткие высказывания в форме «Если-то» и функции принадлежности для соответствующих лингвистических термов выступают в качестве основы для проведения операции нечеткого логического вывода. При этом должно существовать хотя бы одно правило для каждого лингвистического термина выходной переменной и любой терм входной переменной имеет хотя бы одно правило, в котором этот терм используется в качестве предпосылки. Иначе база нечетких правил неполна [34-38].

Функции μ_F соответствуют принадлежности множеству компонентов изображения. Изображения, вводимые в компьютер, часто оказываются что у них изменения яркости малы по сравнению с ее средним значением. При этом яркость меняется не от черного к белому, а от серого к чуть более светлее серого. То есть, реальный диапазон яркости оказывается намного меньше допустимого (шкалы яркости). Задача повышения контраста заключается в «растягивании» диапазона яркости изображения на всю шкалу. Методы обработки изображений на основе теории нечетких множеств предоставляют гибкую математическую базу, для того чтобы справиться с «качественными» свойствами, такими, как контрастность изображения в условиях неоднозначности и расплывчатости, часто присутствующих в цифровых изображениях [36]. С этой целью в [46, 47] предложен алгоритм повышения контрастности и яркости изображений в нечеткой среде. Результат обработки изображения по алгоритму повышения контрастности обозначим через $g_1(x, y)$, а по алгоритму повышения яркости обозначим через $g_2(x, y)$. Рассмотрим устранение третьего фактора —

шума в нечеткой среде. Для сравнения рассмотрим результат обработки изображения среднеарифметическим и среднегеометрическим фильтром.

Рассмотрим зашумленное изображение $z(x, y)$, формируемое прибавлением шума $\eta(x, y)$ к исходному изображению $f(x, y)$:

$$z(x, y) = f(x, y) + \eta(x, y) .$$

Пусть S_{xy} обозначает прямоугольную окрестность, т.е. некоторую область изображения размером $m \times n$ с центром в пикселе (x, y) . Значение яркости пикселя с координатами (x, y) восстановленного изображения представляет собой это значение [26, 29]:

$$g(x, y) = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} z(s, t) .$$

Если на изображении, обработанном среднеарифметическим фильтром с апертурой 3×3 (рис.1.27), регулярные структуры еще различимы, то после обработки фильтром с апертурой 9×9 (рис.1.28) они незаметны, несмотря на значительные перепады яркости на них.

Результат обработки изображения среднегеометрическим фильтром описывается выражением [26, 29]

$$g(x, y) = \left[\prod_{(s,t) \in S_{xy}} z(s, t) \right]^{\frac{1}{mn}} . \quad (1.39)$$

Здесь значение восстановленного изображения в каждом пикселе является корнем степени mn из произведения значений яркости в пикселях окрестности S_{xy} .

Исходное изображение



Шумом типа "соль и перец"



Арифметическое среднее



Рис. 1.27. Арифметическое среднее при размере локального окна 3×3

Исходное изображение



Шум типа «соль и перец»



Арифметическое среднее



Рис. 1.28. Арифметическое среднее при размере локального окна 9×9

На основе выражения (1.39) можно предположить, что результат обработки будет существенно зависеть от размеров локальной апертуры. Это подтверждают обработанные этим методом изображения, которые представлены на рисунках 1.29 и 1.30.

Исходное изображение



Шум типа "соль и перец"



Геометрическое среднее



Рис. 1.29. Геометрическое среднее при размере локального окна 3×3

Исходное изображение



Шум типа "соль и перец"



Геометрическое среднее



Рис. 1.30. Геометрическое среднее при размере локального окна 9×9

Сглаживающие фильтры, как линейные, так и нелинейные, имеют общий недостаток: подавляя верхние частотные компоненты изображения, они не только уменьшают влияние шума, но и приводят к потере мелких объектов изображения и множества тоновых переходов, которые не являются ступенчатыми. Кроме того, процедура упорядочивания пикселей в пределах окна требует значительных вычислительных затрат даже для небольших по размеру изображений. Стандартные фильтры, работающие с окном, ориентированы на неизменные характеристики опорного сигнала, они не могут учитывать изменяющиеся локальные характеристики изображений. Их действие не зависит от характера изображения: текстура это или гладкая цветовая область. Для решения подобной проблемы используется нечеткий подход непосредственно к обработке пикселей окна.

Рассмотрим нечеткое множество $A = \{(z, \mu_A(z))\}$, где Z – дискретное множество элементов $\{z_1, \dots, z_n\}$. При рассмотрении дискретных нечетких множеств и случайных событий следует учитывать, что функция принадлежности задается только на дискретном множестве элементов из Z или R^n . Приведем выражение для нечеткой обработки изображения:

$$g_3(x, y) = \frac{\sum_{i=1}^n \sum_{(s,t) \in S_{xy}} \mu(z_i) z_i(s, t)}{\sum_{i=1}^n \mu(z_i)}.$$

На рис. 1.31 показана нечеткая обработка при размере локального окна 3×3 и 9×9 .

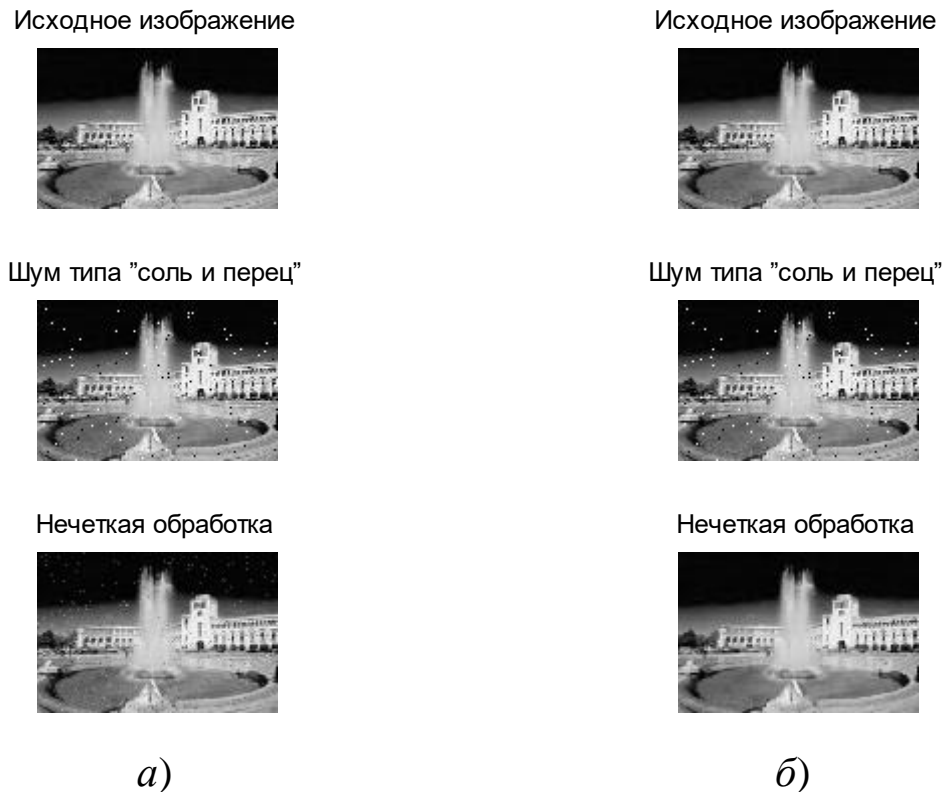


Рис. 1.31. Нечеткая обработка при размере локального окна: *a* - 3×3 ; *б* - 9×9

При создании нечеткой модели количественной оценки качества изображения и обучении параметров нейронными сетями в качестве входных переменных были взяты основные факторы, от которых зависит качество изображения.

1. Яркость представляет собой характеристику, определяющую то, на сколько сильно цвета пикселей отличаются от чёрного

2. Контрастность (Contrast) можно определить как диапазон тонов средней яркости, при увеличении контрастности диапазон яркости средних тонов сужается, а при уменьшении контрастности, наоборот увеличивается, так при увеличении контрастности пиксели, имеющие яркость выше некоторого значения, приобретают еще большую яркость, а пиксели, имеющие яркость ниже некоторого значения, становятся еще темнее;

3. Шум (Noise) — случайные погрешности в каждой точке изображения.

Нечеткий алгоритм адаптивного повышения контрастов изображений

Рассматривается нечеткий алгоритм адаптивного повышения контрастов изображений, который представляет собой характеристику того, насколько большой разброс имеют цвета пикселей изображения. Чем больший разброс имеют значения цветов пикселей, тем больший контраст имеет изображение. Основными недостатками изображений, в большинстве случаев, являются искаженные яркостные характеристики и низкая контрастность.

Яркость представляет собой характеристику, определяющую то, на сколько сильно цвета пикселей отличаются от чёрного цвета. Например, если оцифрованная фотография сделана в солнечную погоду, то ее яркость будет значительной. С другой стороны, если фотография сделана вечером или ночью, то её яркость будет невелика.

Контраст представляет собой характеристику того, насколько большой разброс имеют цвета пикселей изображения. Чем больший разброс имеют значения цветов пикселей, тем больший контраст имеет изображение.

По аналогии с терминами теории вероятностей можно отметить, что яркость представляет собой как бы математическое ожидание значений выборки, а контраст — дисперсию значений выборки.

Яркость и контраст могут рассматриваться не только для всего изображения, но и для отдельных фрагментов. Таким образом, возникают понятия локальной яркости и локального контраста.

Основными недостатками изображений, в большинстве случаев, являются искаженные яркостные характеристики и низкая контрастность.

Поэтому основная цель методов улучшения состоит в преобразовании изображений к такому виду, что делает их более контрастными и, соответственно, более информативнее [11]. Довольно часто на изображении присутствуют искажения в определенных локальных окрестностях, которые вызваны дифракцией света, недостатками оптических систем или розфокусировкой. Это приводит к необходимости выполнения

локальных преобразований на изображении. Иными словами, такой адаптивный подход дает возможность выделить информативные участки на изображении и соответствующим образом их обработать. Изложенным требованиям отвечают методы адаптивного преобразования локального контраста [12]. Для этого использованы такие обозначения:

$F, f(x, y)$ - исходное изображение и его элемент с координатами (x, y) соответственно;

$C(x, y)$ - контраст элемента изображения с координатами (x, y) ;

$G(C(x, y))$ - преобразованное значение контраста $C(x, y)$;

ε, σ, h - характеристики локальных окрестностей (ε - энтропия, σ - среднеквадратичное отклонение, h - функция протяженности гистограммы);

$g(x, y)$ - элемент обработанного изображения с координатами (x, y) .

Недостаток исходного рукописного изображения состоит в том, что это изображение низкоконтрастное, что затрудняет анализ изображения. Поэтому сначала выполняется операция растяжения гистограммы изображения на максимально допустимый диапазон. Далее осуществляется контрастирование исследуемого изображения. Это приводит к улучшению визуального качества рукописных изображений.

Рассмотрим изображение F размера $M \times N$ пиксель, наличие серого уровня r в диапазоне между 0 и $L-1$. При применении нечеткого множества для обработки изображений они рассматриваются как массив нечетких синглтонов. Каждый элемент массива указывает на значение принадлежности $\mu_F(f(x, y))$ серого уровня $f(x, y)$, соответствующее (x, y) -му пикселю, в соответствии с predetermined свойствами изображения, такими как яркость, резкость, однородность [13-16].

В виде обобщения этого подхода введем следующее представление изображения в нечеткой среде.

Изображение F , описанное в нечеткой среде, имеет вид

$$F = \{ \langle f(x, y), \mu_F(f(x, y)) \rangle \mid f(x, y) \in \{0, \dots, L-1\} \},$$

где $x \in \{1, \dots, M\}$, $y \in \{1, \dots, N\}$, $\mu_F(f(x, y))$ обозначают соответственно степень принадлежности (x, y) -го пикселя к множеству в соответствии со свойствами изображения.

Функции μ_F соответствуют принадлежности множеству компонентов изображения. Методы обработки изображений на основе теории нечетких множеств предоставляют гибкую математическую базу, для того чтобы справиться с «качественными» свойствами, такими, как контрастность изображения в условиях неоднозначности и расплывчатости, часто присутствующих в цифровых изображениях.

В терминах нечеткой обработки изображений вопрос, который, естественно, возникает при попытке определить нечеткое множество яркости пикселей, можно сформулировать так: «как мы можем определить принадлежность функции серых уровней для описания изображения в нечетком множестве или более понятным с человеческим восприятием: «как яркое серое значение и как мы можем быть уверены, что он такой яркий?»».

Неопределенность в изображениях исходит из различных факторов. Они влияют на нашу уверенность в принятии решения, является ли пиксель «серым» или «резким», и поэтому вводят определенные сомнения, связанные с соответствующей точкой.

Алгоритм адаптивного повышения контрастности изображений.

1. Нормализация:

$$u(x, y) = l \frac{f(x, y) - f_{\min}}{f_{\max} - f_{\min}}.$$

2. Фаззификация:

$$\mu_F^i(x, y) = \frac{1}{1 + \frac{u(x, y) - c_i}{\sigma_f}}, \quad i = \overline{1, k}.$$

3. Уточнение фаззификации:

$$\mu_F^i(x, y) = \begin{cases} 2(\mu_F^i(x, y))^2, & 0 \leq \mu_F^i(x, y) \leq \frac{1}{2}, \\ 1 - 2(1 - \mu_F^i(x, y))^2, & \frac{1}{2} < \mu_F^i(x, y) \leq 1. \end{cases}$$

4. Определяется нечеткая гистограмма цифрового изображения последовательностью $h_F(f)$ с $f \in \{0, \dots, L-1\}$:

$$h_F(f) = \left\| \left\{ \langle (x, y), \mu_F(f) \rangle \mid x \in \{1, \dots, M\}, y \in \{1, \dots, N\} \right\} \right\|,$$

где $\|\cdot\|$ обозначает число элементов в нечетком множестве.

Более того, $h_F(f)$ представляет собой частоту появления уровня яркости «примерно g ». Однако, из-за его определения, нечеткой гистограмме не удастся быть функцией плотности вероятности.

5. Определяется нормированная гистограмма:

$$\tilde{h}_F(f) = \frac{h_F(f)}{\sum_{f=0}^{L-1} h_F(f)},$$

где $f \in \{0, \dots, L-1\}$.

6. Пусть f - сингльтон, который определяет дискретную интенсивность изображения, $h_F(f(x, y))$ - соответствующие значения гистограммы. n -й момент $f(x, y)$ относительно среднего значения яркостей элементов локальной окрестности W . определяется формулой:

$$M_n(f) = \sum_{(x,y) \in W} \left(f(x, y) - \frac{\sum_{i=1}^k f_i(x, y) \cdot \mu_F^i(x, y)}{\sum_{i=1}^k \mu_F^i(x, y)} \right)^n h_f(f(x, y)).$$

7. Определяется мера контраста на основе следующего выражения:

$$C(x, y) = 1 - \frac{1}{1 + k \frac{\sum_{j=1}^n [f_j - M[f_j]]^2 \mu_j}{\sum_{j=1}^n \mu_j}}, \quad (1.40)$$

$C(x, y)$, согласно выражению (1.40), равно нулю для окрестностей с постоянной интенсивностью и единице - для больших значений $\sigma^2(L)$. Это свойство выражения (1.40) полностью отвечает требованиям определения локального контраста.

8. Мера нечеткости F может быть определена по аналогии с энтропией Шеннона в виде [4]:

$$\varepsilon(\mu_F) = -a \sum_{i=1}^n \{ \mu_F(f_i) \ln \mu_F(f_i) + [1 - \mu_F(f_i)] \ln [1 - \mu_F(f_i)] \},$$

9. Осуществляется нелинейное преобразование локального контраста $C(x, y)$.

$$C^*(x, y) = C(x, y)^{\alpha_{\min} + (\alpha_{\max} - \alpha_{\min})(\varepsilon(x, y) - \varepsilon_{\min}) / (\varepsilon_{\max} - \varepsilon_{\min})^s}.$$

Здесь задаются минимальное α_{\min} и максимальное α_{\max} значения показателя степени α . Адаптация состоит в формировании дополнительного слагаемого к α_{\min} путем его определения на основе энтропия ε значений яркостей элементов скользящей окрестности.

10. Восстанавливается изображение путем определения нового значения яркости $g(x, y)$ элемента с координатами (x, y) . Для этого используем выражение, которое определяется из формулы (1.40):

$$g(x, y) = \bar{f}(x, y) + \left(\frac{C^*(x, y) * n * m}{1 - C^*(x, y)} - \sum_{\forall (x, y) \in W_2 - W_1} \left(\frac{\sum_{i=1}^k f_i(x, y) \cdot \mu_F^i(x, y)}{\sum_{i=1}^k \mu_F^i(x, y)} - f(x, y) \right)^2 h_F(f(x, y)) \right)^{0.5}.$$

Описанную процедуру повторяем для каждого элемента изображения.

В предложенном методе используется статистическое определение локальных контрастов, благодаря чему учитываются такие характеристики текстуры, как однородность, шершавость и зернистость. Поэтому данный метод рекомендуется применять для обработки изображений, которые содержат мелкие детали (рис. 1.32).



a)



б)



в)



г)

Рис. 1.32. Обработка изображений методом степенного преобразования локальных контрастов с использованием известных и статистических выражений их определения:
 a - $\alpha = 1$; b - $\alpha = 0.5$; v - $\alpha = 0.1$; z - $\alpha = 0.01$

Предложенное выражение для модифицированного степенного преобразования позволяет более четко идентифицировать различные типы локальных окрестностей изображения и адаптивно усиливать их контраст в зависимости от значений локальных характеристик этих окрестностей.

Метод усиления контраста эффективно используется в обработке широкого класса изображений. Учитывая характеристики скользящих окрестностей удается идентифицировать участки изображения по уровню контрастности и соответствующим образом на них реагировать. Благодаря этому достигается более тонкая обработка мелких деталей. Однако изображения должны отвечать двум требованиям. Они не должны содержать большого количества импульсных выбросов и темные или светлые участки большой площади. Ведь в первом случае это может привести к неадекватному вычислению функции протяженности гистограммы, а во втором - к неэффективному

Алгоритм нелинейных преобразований локальных контрастов

Усиление контраста - одна из важных задач обработки изображений, распознавания образов и машинного зрения. Решение этой задачи непосредственно связано с повышением вероятности правильного восприятия изображения.

Аналитическим выражением, описывающим количественное определение реакции зрительной системы на световое возбуждение, является его контраст. Вид этого выражения определяется свойствами конкретной зрительной системы восприятия [11,12]. То есть изменение выражения определения контраста отвечает изменению типа зрительной системы или ее параметров. Это создает возможности адаптации зрительной системы путем изменения выражения для определения локального контраста. Разумеется, что при этом аналитическое выражение должно обеспечивать сохранение основных предельных свойств. Эти свойства заключаются в том, что локальный контраст $C(x, y)$ приобретает максимальное значение только тогда, когда его компоненты имеют значения, которые лежат на противоположных краях диапазона, и равный нулю - в случае равенства этих компонентов по величине. Критерием оценки выражений контраста

является эффективность их применения при цифровой обработке изображений. Следовательно, удачный выбор того или иного определения контраста существенно влияет на дальнейшее применение метода.

Качество изображения на локальных участках можно улучшать, используя такие параметры интенсивностей пикселей, как среднее значение интенсивности и изменение интенсивности (или стандартное среднеквадратичное отклонение интенсивностей элементов локальной окрестности изображения). Среднее значение - это мера средней яркости. При вычислении и анализе средней яркости элементов изображения существует возможность ее коррекции, т.е. затемненные участки изображения делать более светлыми, а слишком светлые участки изображения затемнять. Однако в случае, если на изображении присутствуют темные и светлые области, то такой подход только ухудшит его визуальное восприятие. Поэтому целесообразно использовать еще один параметр, который характеризовал бы распределение яркостей элементов изображения в некоторой локальной окрестности. Другими словами, этот параметр характеризовал бы изменения интенсивностей или меру контрастности изображения.

Усиление контраста - одна из важных задач обработки изображений, распознавания образов и машинного зрения. Решение этой задачи непосредственно связано с повышением вероятности правильного восприятия изображения. В последнее время разработаны нечеткие методы улучшения изображений, которые базируются на нелинейных преобразованиях локальных контрастов с учетом особенностей человеческого зрения [13,14]. Изображение F , описанное в нечеткой среде, имеет вид:

$$F = \{ \langle f(x, y), \mu_F(f(x, y)) \rangle \mid f(x, y) \in \{0, \dots, L-1\} \},$$

где $x \in \{1, \dots, M\}$, $y \in \{1, \dots, N\}$, $\mu_F(f(x, y))$ обозначают соответственно степень принадлежности (x, y) -го пикселя к множеству в соответствии со свойствами изображения.

Алгоритм нелинейных преобразований локальных контрастов:

Реализация этих методов состоит в выполнении следующих основных шагов:

Шаг 1. Определение количественной меры локального контраста.

Для описания контраста 8-битного полутонового цифрового изображения предложены два варианта формул вычисления локального контраста:

$$C(x, y) = (C_{\max} - C_{\min}) / 255$$

и вычисление глобального контраста:

$$C(x, y) = \left(\frac{\sum_{j=1}^n [f_j - M[f_j]]^2 \mu_j}{\sum_{j=1}^n \mu_j} \right)^{0.5} / 255,$$

где C_{\max} , C_{\min} – максимальное и минимальное значения яркости в окрестностях пикселей.

Рассмотрим несколько окрестностей, которым присуща различная степень гладкости:

а) локальная окрестность с одинаковыми уровнями яркостей (однородная окрестность);

б) локальная окрестность, элементы которой имеют значения яркостей, находящиеся на противоположных концах диапазона (условно бинарная окрестность);

в) локальная окрестность, которая содержит элементы, значения яркостей которых не являются одинаковыми и не находятся на краях диапазона.

Приведенные выше типы окрестностей будут характеризоваться различными значениями локальных характеристик [15,16]. Рассмотрим это более детально на примере энтропии, функции протяженности гистограммы и среднеквадратичного отклонения.

Шаг 2. Определение функция протяженности гистограммы, которая вычисляется по выражению

$$h_F(x, y) = \frac{f_{\max} - f_{\min}}{h_{\max}},$$

где f_{\min} , f_{\max} – минимальное и максимальное значения яркости в скользящей окрестности W с центром в элементе с координатами (x, y) ;

h_{\max} – максимальное значение гистограммы скользящей локальной окрестности W с центром в элементе с координатами (x, y) .

Эта характеристика локальной окрестности приобретает минимальные значения на однородных участках, и максимальные - на бинарных участках.

Шаг 3. Определение степени преобразования локальных контрастов α по функциям протяженности гистограммы:

$$\alpha = (\alpha_{\min} - \alpha_{\max}) \left(1 - \exp\left(-\frac{(h_F - a)^2}{2\pi^2}\right) \right)^s,$$

где $s > 0$.

Шаг 4. Определение нечеткой энтропии в скользящей локальной окрестности размерами $n \times m$ по выражению:

$$\varepsilon(\mu_F) = -a \sum_{i=1}^n \{ \mu_F(f_i) \ln \mu_F(f_i) + [1 - \mu_F(f_i)] \ln [1 - \mu_F(f_i)] \} / \log(nm), \quad (1.41)$$

где $\mu_F(f_i)$ вычисляется следующим образом:

$$\mu_F(f_i) = h_F(f_i(x, y)) / (n \times m).$$

Здесь: $h_F(f_i(x, y))$ - значение гистограммы локальной окрестности W (количество элементов с яркостью $f_i(x, y)$ в окрестности W) для величины яркости элемента с координатами (x, y) .

Согласно выражению (1), нечеткая энтропия приобретает максимальное значение на однородных участках, а минимальное - на участках с элементами, значения яркостей которых находятся на противоположных краях диапазона.

Шаг 5. Определение степени преобразования локальных контрастов α по нечеткой энтропии

$$\alpha = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \left(\frac{\varepsilon(\mu_F) - \varepsilon_{\min}}{\varepsilon_{\max} - \varepsilon_{\min}} \right)^s,$$

где $s > 0$.

Шаг 6. Определение среднеквадратичного отклонения значений яркостей элементов скользящей окрестности W , которое вычисляется выражением

$$\sigma(x, y)_F = \sqrt{\frac{1}{nm} \left(\frac{\sum_{j=1}^n [f_j - M[f_j]]^2 \mu_j}{\sum_{j=1}^n \mu_j} \right)}, \quad (1.42)$$

где $M[f_j]$ - нечеткое среднеарифметическое значение яркостей элементов локальной окрестности W_c с центром в элементе $M[f_j(x, y)]$ с координатами (x, y) :

$$M[f_j] = \frac{1}{NM} \sum_{x=1}^N \sum_{y=1}^M f_j(x, y),$$

здесь N, M - размеры изображения ($x = \overline{1, N}, y = \overline{1, M}$),

Выражение (1.41) равно нулю для однородных окрестностей и возрастает с увеличением неоднородности.

Шаг 7. Определение степени преобразования локальных контрастов α по нечеткому среднеквадратичному отклонению значений яркостей:

$$\alpha = \alpha_{\min} \sigma(x, y) + \alpha_{\max} (1 - \sigma(x, y))^s$$

где $s > 0$

Шаг 8. Увеличение по определенному закону некоторой количественной меры локального контраста.

Для нелинейного преобразования локального контраста используем следующее выражение:

$$C^*(x, y) = \begin{cases} B_0 + \left(\frac{R}{2} - A_0 \right) \left(\frac{C(x, y) - C_{\min}}{\hat{C} - C_{\min}} \right)^\alpha & C(x, y) \leq \hat{C}, \\ R - A_0 - \left(\frac{R}{2} - A_0 \right) \left(\frac{C_{\max} - C(x, y)}{C_{\max} - \hat{C}} \right)^\alpha & C(x, y) > \hat{C}, \end{cases}$$

где

$C(x, y)$ - значение локального контраста элемента исходного изображения с координатами (x, y) ,

$C^*(x, y)$ - усиленное значение локального контраста элемента изображения с координатами (x, y) ;

R - максимально возможное значение локального контраста $R = I$;

C_{\min}, C_{\max} - максимальное и минимальное значения локального контраста исходного изображения;

\hat{C} - оценка математического ожидания значений локального контраста (например, среднеарифметическое значение локальных контрастов элементов изображения);

A_0, B_0 - коэффициенты постоянного смещения;

α - показатель степени ($\alpha < 1$).

Шаг 9. Восстановление элемента преобразованного изображения с усиленным локальным контрастом.

Теперь, когда известен характер изменения значений локальных статистик в скользящих окрестностях различного типа, рассмотрим процедуру формирования функции преобразования локальных контрастов. Важная проблема, которая стоит перед исследователем при формировании этой функции, заключается в том, насколько нужно увеличить локальные контрасты на том или ином участке изображения. Характер изменения усиления определяется локальными статистиками, но границы изменения ($\alpha_{\min}, \alpha_{\max}$) задаются исследователем. Это связано с тем, что пока не существует теоретического решения проблемы оптимальности преобразования локального контраста. Поэтому, исходя из опыта и знаний исследователя, функцию преобразования формируют таким образом, чтобы она обеспечивала максимальную контрастность изображения при минимуме искажений, вызванных чрезмерным усилением локальных контрастов.

Приводим результаты работы программы методом нелинейного растяжения локальных контрастов с изображениями, видим результирующий эффект и возможное практическое применение (рис. 1.33–1.35).

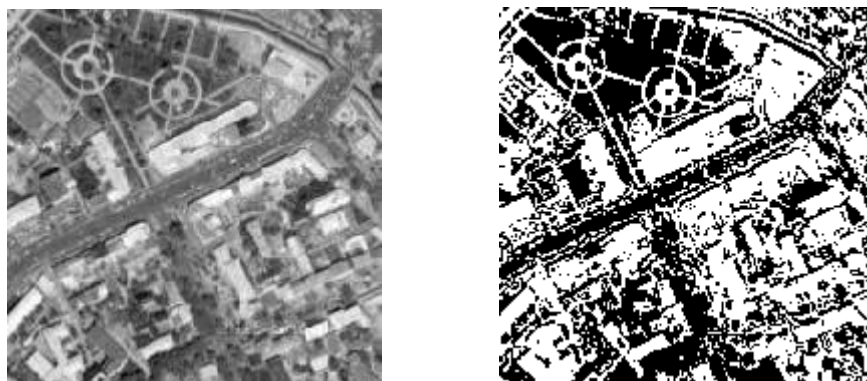


Рис.1.33. Обработка первого изображения методом нелинейного растяжения локальных контрастов

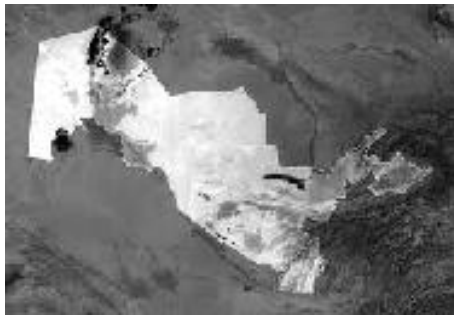


Рис.1.34. Обработка второго изображения методом нелинейного растяжения локальных контрастов



Рис.1.35. Обработка третьего изображения методом нелинейного растяжения локальных контрастов

Отметим, что при программной реализации предложенного метода учитывают случай, когда $\sigma_F(x, y) = 0$, задавая некоторое предельное минимальное значение σ_{\min} . То есть текущему среднеквадратичному отклонению значений яркостей элементов изображения $\sigma_F(x, y)$ присваивают σ_{\min} в том случае, когда $\sigma_F(x, y) < \sigma_{\min}$



Рис.1.36. Локально-адаптивный метод улучшения визуального качества изображений с использованием среднеквадратичных отклонений значений яркостей элементов локальной окрестности:
а - исходное изображение; *б* - изображение *а*, обработанное известным методом; *в* - изображение *а*, обработанное предложенным методом с использованием теории нечетких множеств

Таким образом, этот метод может быть реализован в скользящем варианте, в котором преобразование локальных контрастов выполняется с учетом характеристик локальных окрестностей.

Важным фактором эффективного применения адаптивных методов является правильный выбор функции адаптивного преобразования локальных контрастов. В данной работе рассматриваются только степенные функции преобразования типа $G(C) = C^{\alpha_{адап}}$. Далее будем более детально рассматривать только степень преобразования локальных контрастов α . При формировании таких функций задают минимальное (α_{\min}) и максимальное (α_{\max}) значения степени α , причем $\alpha_{\min} > 0, \alpha_{\max} < 1$. А сама адаптация состоит в формировании дополнительного слагаемого к α_{\min} на основе некоторой нечеткой статистики (энтропия, функция протяженности гистограммы, среднеквадратичное отклонение). Функции преобразования должны удовлетворять условиям:

$$C(i, j) \in [0,1], \quad F(C(i, j)) \geq C(i, j), \quad F(C(i, j)) \in [0,1].$$

Отметим, что выбор той или иной функции преобразования зависит от того, какая статистика используется для характеристики гладкости локальной окрестности W .

Алгоритм нелинейных преобразованиях гистограммы локальных контрастов

Метод усиления контраста с использованием функции протяженности гистограммы эффективно используется в обработке широкого класса изображений.

Учитывая характеристики скользящих окрестностей, удастся идентифицировать участки изображения по уровню контрастности и соответствующим образом на них реагировать. Благодаря этому достигается более тонкая обработка мелких деталей. Однако изображения должны отвечать двум требованиям. Они не должны содержать большого количества импульсных выбросов и темные или светлые участки большой площади. Ведь в первом случае это может привести к неадекватному вычислению функции протяженности гистограммы, а во втором - к неэффективному усилению контраста.

Нечеткая гистограмма цифрового изображения является последовательностью $h_F(f)$ с $f \in \{0, \dots, L-1\}$ и определяется как

$$h_F(f) = \left\| \left\{ \langle (x, y), \mu_{\tilde{f}_{ij}}(f) \rangle \mid i \in \{1, \dots, M\}, j \in \{1, \dots, N\} \right\} \right\|,$$

где $\|\cdot\|$ обозначает число элементов в нечетком множестве.

Более того, $h_F(f)$ представляет собой частоту появления уровня яркости «примерно f ». Однако, из-за его определения, нечеткой гистограмме не удастся быть функцией плотности вероятности.

Рассмотрев гистограммы распределения значений локальных контрастов реальных изображений, отметим, что в большинстве случаев они имеют небольшие значения и занимают примерно треть допустимого диапазона. Следовательно, реальные

изображения характеризуются, в основном, небольшими значениями локальных контрастов.

Рассмотрим алгоритм повышения качества изображения, который базируется на адаптивном преобразовании локального контраста.

Алгоритм повышения качества изображения. Адаптация в данном методе осуществляется на основании анализа такой характеристики, как функция протяженности гистограммы элементов локальной скользящей окрестности. Для примера будем считать, что элементы изображения представлены 8-разрядными целыми числами, т.е. $f(x, y) \in [0, 255]$.

Основные шаги реализации этого алгоритма следующие.

Шаг 1. Определяем характеристику локальной скользящей окрестности, используя функцию протяженности гистограммы:

$$h(W(x, y)) = \frac{f_{\max}(W(x, y)) - f_{\min}(W(x, y))}{h_{\max}(W(x, y))}, \quad (1.43)$$

где $f_{\max}(W(x, y)), f_{\min}(W(x, y))$ - соответственно максимальное и минимальное значения яркостей элементов скользящей окрестности с центром в элементе с координатами (x, y) ; $h_{\max}(W(x, y))$ - максимальное значение гистограммы уровней яркости элементов окрестности с центром в элементе с координатами (x, y) .

Шаг 2. Нормированная гистограмма имеет вид:

$$\tilde{h}_F(f) = \frac{h_F(f)}{\sum_{f=0}^{L-1} h_F(f)},$$

где $f \in \{0, \dots, L-1\}$.

Сопоставление нормированных гистограмм возможно, если для них совпадает носитель и число ячеек. Тогда можно выстроить меру схожести нормированных гистограмм, например, как меру Хэмминга. Тогда предельно схожие гистограммы имеют меру схожести 0, а предельно расходящиеся нормированные гистограммы – единицу (т.е. носитель критерия – стандартный 01-носитель).

Шаг 3. Определение критерий схожести. Можно получить лингвистическую интерпретацию в виде пенташкалы «ОН – Н – Ср – В – ОВ». Построить такую пенташкалу можно, если на систематической основе давать экспертному сообществу на

сопоставление две нормированные гистограммы с предложением дать лингвистическую оценку схожести этих гистограмм.

Наносим некоторое количество узловых точек (три или сразу пять). Если нет никаких мыслей относительно положения узловых точек (нет эксперта под рукой) - работает стандартное правило: узловая точка ОН – левый конец интервала носителя, узловая точка ОВ – правый конец интервала носителя, средняя точка (Ср) – отвечает максимуму гистограммы (в унимодальном случае) или медианегистограммы (в полимодальном случае). В принципе, можно везде пользоваться медианой вместо среднего (для нормального распределения эти величины совпадают).

Есть исключение из общего правила предыдущего пункта, когда у гистограммы максимум прижат к левой или правой точке интервала носителя. Такая ситуация говорит о том, что существует определенная тенденция, при которой максимум гистограммы не соответствует среднему уровню параметра.

Шаг 4. Построим набор из пяти узловых точек пятиуровневого классификатора по правилу:

$$\begin{aligned}f_1 &= f - t_1\sigma, \\f_2 &= f - t_2\sigma, \\f_3 &= f - t_3\sigma, \\f_4 &= f + t_2\sigma, \\f_5 &= f + t_1\sigma,\end{aligned}$$

Для каждой узловой точки классификатора справедливо, что в ней уровень фактора распознается, однозначно, со стопроцентной экспертной уверенностью. Например, точка f_1 отвечает очень низкому уровню фактора (ОН), f_2 – состоянию Н и т.д.

Шаг 5. Если мы выбрали только три узловые точки из пяти, то оставшиеся две мы наносим по правилам взвешенного среднего. Таким образом, мы учитываем при выборе точек F гистограммы.

Интервал между двумя рядом стоящими узловыми точками мы делим на три зоны, промежуточная из которых – это зона неуверенности эксперта в классификации, интерпретируемая наклонным ребром трапецевидного нечеткого числа. Таким образом, первичная лингвистическая интерпретация гистограммы завершена.

Таким образом, поделим каждый отрезок $[f_i, f_{i+1}]$ на три зоны: зону абсолютной уверенности, зону пониженной уверенности и зону абсолютной неуверенности. Длины этих трех зон составляют пропорцию $1:u:1$, где параметр $u \geq 0$ выражает глубину неуверенности. Так, при $u=0$ пониженной уверенности нет, и разграничение зон является жестким (интервальным). В противоположном случае, при $u=\infty$, абсолютной уверенности-неуверенности нет.

Для носителя $F=[0, 10]$ определяем: $f = 4.5$, СКО = 2. Также задаемся $u=1$, т.е. все зоны уверенности - пониженной уверенности - неуверенности имеют равную длину.

Значение 0.5 носителя гистограммы представляется экспертам очень низким, а значение 8.5 – очень высоким. Отсюда и из (4) сразу следует $t_1 = (4.5 - 0.5)/2 = (8.5 - 4.5)/2 = 2$. Значение 2 носителя представляется экспертам низким, поэтому $t_2 = (4.5 - 2)/2 = 1.25$. Соответственно, непротиворечивая классификация дает $f_4 = f + t_2\sigma = 4.5 + 1.25 * 2 = 7$.

Пусть эксперты единодушно договорились о том, чтобы считать:

$$f_1 = 50$$

$$f_3 = 100$$

$$f_5 = 255$$

$$u = 1.$$

Но эксперты затрудняются с получением узловых точек f_2 и f_4 . Можно было бы примитивно задать $f_2 = (f_1+f_3)/2 = 100$, $f_4 = (f_3+f_5)/2 = 225$, но такое задание пренебрегает формой гистограммы, а ее хотелось бы учесть в анализе. Такой учет возможен, если искать узловые точки по правилам взвешенного среднего. Тогда:

$$f_2 = \frac{50 * \tilde{h}_2 + 75 * \tilde{h}_3 + 100 * \tilde{h}_4 + 125 * \tilde{h}_5 + 150 * \tilde{h}_6}{\tilde{h}_2 + \tilde{h}_3 + \tilde{h}_4 + \tilde{h}_5 + \tilde{h}_6};$$

$$f_4 = \frac{150 * \tilde{h}_6 + 175 * \tilde{h}_7 + 200 * \tilde{h}_8 + 225 * \tilde{h}_9 + 240 * \tilde{h}_{10} + 245 * \tilde{h}_{11} + 255 * \tilde{h}_{12}}{\tilde{h}_6 + \tilde{h}_7 + \tilde{h}_8 + \tilde{h}_9 + \tilde{h}_{10} + \tilde{h}_{11} + \tilde{h}_{12}}.$$

Шаг 6. Вычисляем локальный контраст элемента.

Шаг 7. Вычисляем степенное преобразование локального контраста, которое благодаря использованию функции протяженности гистограммы скользящей окрестности имеет адаптивный характер:

$$C^*(x, y) = C(x, y)^\alpha,$$

где

$$\alpha = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \frac{h(W(x, y)) - h_{\min}(W(x, y))}{h_{\max}(W(x, y)) - h_{\min}(W(x, y))},$$

$h_{\max}(W(x, y))$, $h_{\min}(W(x, y))$ - соответственно максимальное и минимальное значения функции протяженности гистограммы для окрестности с центром в элементе с координатами (x, y) .

Шаг 8. Восстанавливаем элемент преобразованного изображения с усиленным контрастом.

Шаг 9. Оценка возможные значения функции протяженности гистограммы h , скользящей окрестности W , где изображениям присущи три характерных типа окрестностей.

Первый тип - это однородный участок изображения, который характеризуется примерно одинаковыми уровнями яркостей элементов: $f_{\max} \approx f_{\min}$, а следовательно, согласно выражению (1.43), функция протяженности гистограммы локальной окрестности $h(W(x, y))$ будет равна нулю. Локальные контрасты таких участков изображения усиливать не нужно.

Для бинарных участков изображения с примерно одинаковым количественным соотношением элементов f_{\min} и f_{\max} в скользящей окрестности W . Предполагая, что для темных и светлых элементов бинарной окрестности с примерно равным количественным соотношением максимальное значение гистограммы будет равно

$$h_{\max}(W(x, y)) = \frac{mn}{2},$$

где m и n - размеры скользящей окрестности W , выражение (1.43) примет вид

$$h(W(x, y)) = \frac{f_{\max}(W(x, y)) - f_{\min}(W(x, y))}{mn/2}.$$

Третьим характерным типом возможной локальной окрестности является такая окрестность, где в примерно

одинаковой мере присутствуют элементы со всеми возможными яркостями с диапазона $[0,255]$. Такие окрестности характеризуются гистограммой равномерного распределения яркостей. Для такой окрестности будем считать, что она высококонтрастна и не нуждается в усилении контраста.

Выше были рассмотрены граничные случаи локальных окрестностей. Все другие окрестности характеризуются такими значениями функций протяженности гистограммы, которые находятся в диапазоне $[0,255]$.

На основании анализа рассмотренных типов окрестностей и соответствующих им значений функций протяженности гистограммы можно более объективно подойти к формированию степенной функции преобразования локального контраста. Наиболее удобно такой анализ проводить с помощью графического представления функции преобразования локального контраста. Укажем при этом, что $0 < \alpha < 1$ и уменьшение α отвечает более высокому усилению локального контраста, а увеличение - более слабому его усилению.

Максимальное усиление локального контраста испытывают однородные участки изображения ($h_s \approx 0$), что не всегда желательно. Ведь однородные участки очень чувствительны к помехам, поэтому чрезмерное усиление их контраста приведет к значительным искажениям. Экспериментальные исследования показывают, что максимальному усилению ($C^{\alpha_{\min}}$) должны подвергаться локальные контрасты в таких скользящих окрестностях, для которых функция протяженности гистограммы приобретает значения с середины диапазона $h_s \in [0,255]$.

В соответствии с изложенными требованиями нами предлагается использовать степенную функцию преобразования локального контраста, характер изменения показателя степени такое:

$$\alpha = \alpha_{\min} + (a_{\max} - \alpha_{\min}) \left[1 - \exp\left(-\frac{(h_s - a)}{0,0392}\right) \right]^s, \quad (1.44)$$

где α - значение функции протяженности гистограммы, которое отвечает наиболее информативным участкам изображения ($\alpha \approx 0,5$); s - постоянный коэффициент ($s > 1$).

Предложенное выражение (1.44) для модифицированного степенного преобразования позволяет более четко идентифицировать различные типы локальных окрестностей изображения и адаптивно усиливать их контраст в зависимости от значений локальных характеристик этих окрестностей.

Можно пытаться уточнять полученную классификацию, сближая узловые точки классификации друг с другом и сужая тем самым зону неопределенности. Можно заместить узловую точку интервалом абсолютной уверенности и пытаться его расширять в обе стороны от узловой точки, которая этому интервалу принадлежит по построению. Но во всех случаях мы должны обеспечивать согласие экспертного сообщества о результатах уточненной классификации.

Если гистограмма невыразительна (имеет невыраженный максимум и множество локальных минимумов), то наиболее рациональным является переход от исходного интервала носителя к 01-интервалу (путем линейного преобразования масштаба), выстраивание на 01-носителе стандартной симметричной пятиуровневой классификации и последующий перенос этой классификации на исходный носитель (обратное масштабирование путем линейного преобразования). Что такое невыраженный максимум – это такая же категория, как и «счастье», которое каждый склонен понимать по-своему.

Так или иначе, лингвистическая классификация доступна только опытным экспертам, хорошо понимающим природу объекта исследования и действующие в отношении этого объекта закономерности. Приемы, описанные в предыдущих пунктах изложения, являются вспомогательными. Они облегчают жизнь экспертам, но не являются правилом на все случаи жизни.

Влияние ошибок квантования можно увидеть путем сравнения четких и нечетких гистограмм. В случае «жесткого» первого порядка статистики существует ряд уровней серого с нулевой или почти нулевой частотой появления из-за шумов квантования, в то время как уровни серого в их окрестности обладают высокой частотой [25]. Это не тот случай, когда рассматривается нечеткая гистограмма. Поэтому для того чтобы моделировать неопределенности родом из шума квантования, колебания, соответствующие уровню серого изображения, должны быть

пропорциональны нормированной абсолютной разнице между нормализованной четкой и нечеткой гистограммой.

Шаг 10. Комбинируя все приведенные выше утверждения, смоделируем гистограмму $\pi_F(f)$, соответствующую серому уровню f изображения F :

$$\pi_F(f) = (1 - \mu_F(f)) \frac{|\tilde{h}_c(f) - \tilde{h}_F(f)|}{\max\{|\tilde{h}_c(f) - \tilde{h}_F(f)|\}}, \quad (1.45)$$

где $\tilde{h}_c(f)$ является нормированной гистограммой четких изображений.

Размытость изображения - это мера серости двусмысленности, связанная с пикселями изображения. Иногда требуется в несколько раз уменьшить размытость, присутствующую в изображении, в целях повышения контрастности между яркой и темной областями.

$$\mu_{\tilde{F}}(f) = \left(1 + \frac{f_{\max} - f}{F_d}\right)^{-F_e},$$

где f_{\max} обозначает максимально серый уровень желаемого, а F_e , F_d являются экспоненциальным и деноминационным фаззификаторами соответственно, которые контролируют неопределенность в нечетких плоскостях.

Фаззификатор F_d определяется как

$$F_d = \frac{f_{\max} + f}{\left(\frac{1}{2}\right)^{\frac{-1}{F_e}} - 1}.$$

Шаг 11. Модификация принадлежности значений осуществляется с помощью оператора интенсификации [15] в следующем виде:

$$T_1(\mu_{\tilde{F}}(f)) = \begin{cases} 2(\mu_{\tilde{F}}(f))^2, & 0 \leq \mu_{\tilde{F}}(f) \leq \frac{1}{2}, \\ 1 - 2(1 - \mu_{\tilde{F}}(f))^2, & \frac{1}{2} \leq \mu_{\tilde{F}}(f) \leq 1. \end{cases}$$

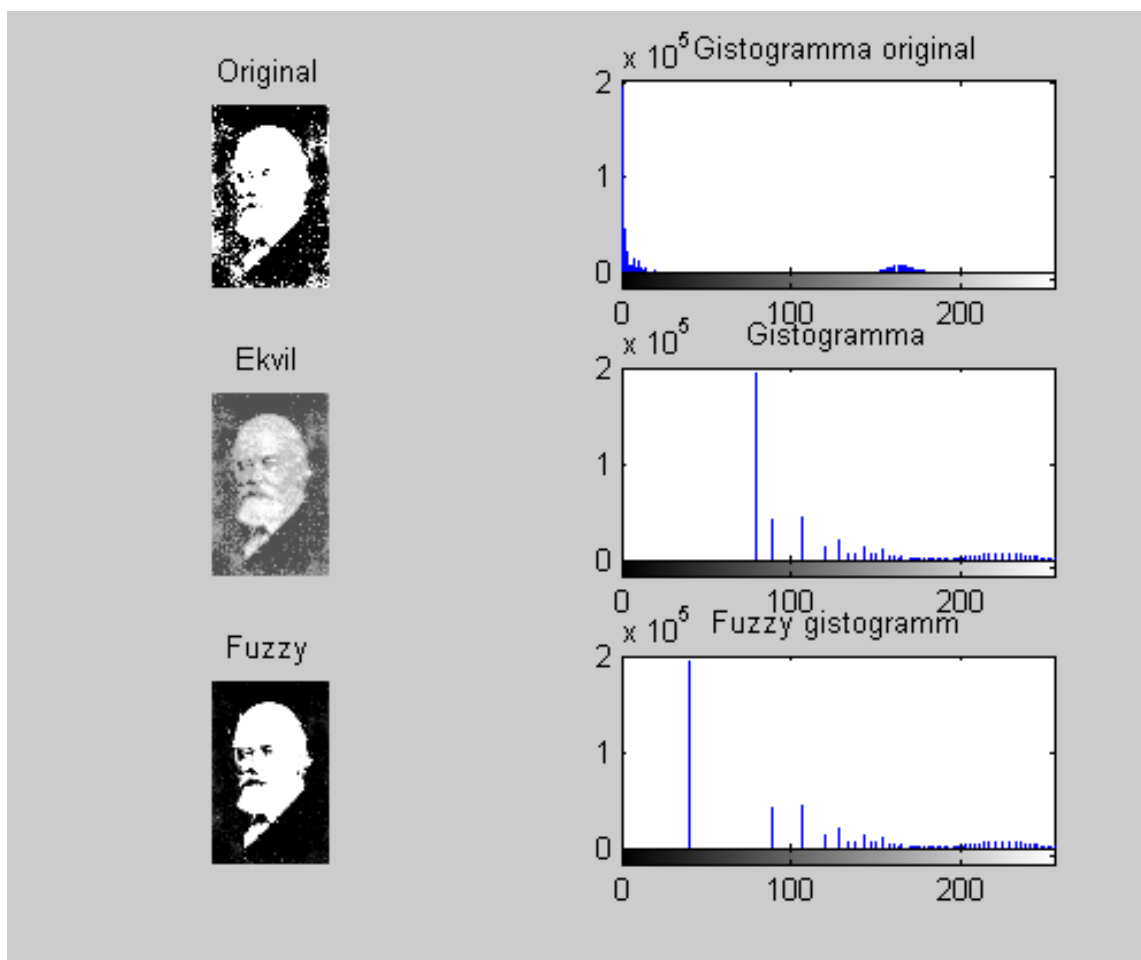
Шаг 12. После модификации значений принадлежности дефаззификация выполняется в следующем виде:

$$f' = \begin{cases} 0, & \bar{f}' < 0, \\ \bar{g}', & 0 \leq \bar{f}' \leq 255, \\ 255, & \bar{f}' > 255, \end{cases}$$

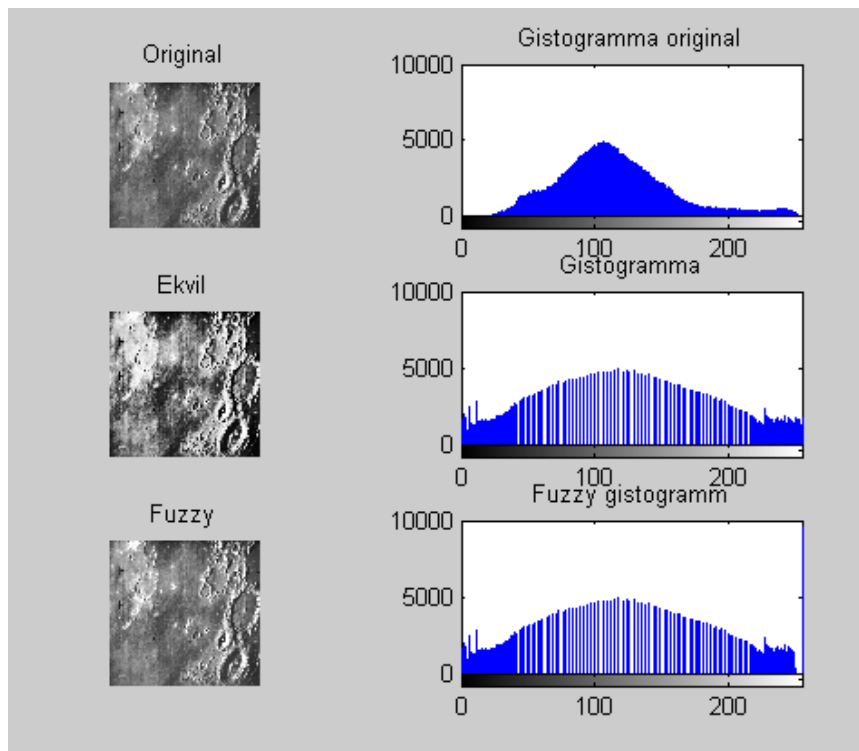
где \bar{f}' новый серый уровень и \bar{f}' получается из обратной функции принадлежности как

$$\bar{f}' = f_{\max} - F_d \left((\mu'_F(f))^{\frac{-1}{F_e}} - 1 \right).$$

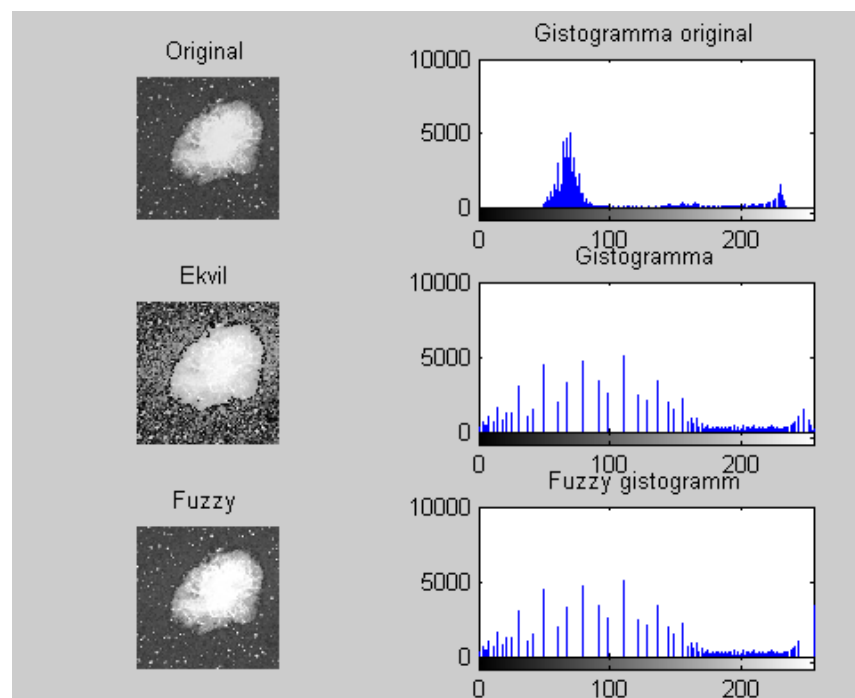
На рис. 1.37 приведены результаты работы рассмотренных алгоритмов на разных изображениях [17-19].



a)



б)



в)

Рис. 1.37. Результаты работы рассмотренных алгоритмов:

Таким образом, при обработке изображений требуется по некоторым признакам выделять некоторые однородные области изображения. Этапы предварительной обработки изображения позволяют уменьшить влияние искажений на процесс распознавания. На основе полученных результатов можно заключить, что предложенный нечеткий подход вводит новый потенциал в решение различных задач обработки изображений, так как он предлагает гибкий и адаптируемый способ обработки неопределенности, присутствующей в цифровых изображениях.

Нейро-нечеткая сеть количественной оценки качества изображения.

При построении нейро-нечеткой сети количественной оценки качества изображения для лингвистической оценки каждой переменной использовались три термина — «плохой — bad», «хороший — good», «отличный — excellent». Для выходной переменной, которая обозначает качество изображения (quality), используем пять термов (плохой — bad, лучше — worse, хороший — good, более подходящий — better, отличный — excellent) с гауссовыми функциями принадлежности. При составлении логических правил учитывалось, что качество изображения ухудшается при большом наличии шумов, плохой яркости и контрастности. При отсутствии этих факторов или их снижении качество изображения улучшается. Модель оценки качества изображения основывается на нечетком логическом выводе Сугено, в котором все значения входных и выходных переменных заданы нечеткими множествами. Базу знаний Сугено можно трактовать как разбиение пространства влияющих факторов на подобласти с размытыми границами, внутри которых функция отклика принимает нечеткое значение.

Нейро-нечеткая сеть количественной оценки качества изображения исходного изображения и полученного после обработки состоит из шести слоев и нулевой слой, на который поступает входной сигнал в виде матрицы, столбцы которой соответствуют разным классам обрабатываемого изображения. В нулевом слое этот сигнал преобразуется в вектор с элементами q_i^k , где $i = \overline{1, N \times M}$, k принимает значения — от нуля до 3 в нашей задаче, т.е. $q_i^0 = f$, $q_i^1 = g_1$, $q_i^2 = g_2$, $q_i^3 = g_3$ соответственно, а $N \times M$ определяет число одновременно обрабатываемых пикселей [29].

1. В первом слое осуществляется фаззификация каждой компоненты входного сигнала q_i^k с помощью набора функций принадлежности $\mu_{ji}^k(q_i^k)$, $j=1,2,\dots,m$, в нашем случае – гауссовы функции вида

$$\mu_{ji}^k(q_i^k) = \exp\left(-\frac{1}{2}\left(\frac{q_i^k - c_j^k}{\sigma_j^k}\right)^2\right),$$

где c_j^k определяет положение центра j -й функции принадлежности k -го класса, а σ_j^k – ее ширину.

2. Во втором скрытом слое производится агрегирование уровней принадлежности входных сигналов по каждому классу на основе вычисления нечеткой T -нормы вида

$$w_j^k = \prod_{l=1}^{N \times M} \mu_{jl}^k(q_l^k).$$

3. В третьем слое реализуется нормализация:

$$\bar{w}_j^k = \frac{w_j^k}{\sum_{j=1}^m \prod_{l=1}^{N \times M} \mu_{jl}^k(q_l^k)},$$

где

$$\sum_{j=1}^m \bar{w}_j^k = 1.$$

4. В четвертом скрытом слое рассчитывается m функций консеквента

$$r_j^k(q^k) = b_{j0}^k + \sum_{l=1}^{N \times M} b_{jl}^k q_l^k \quad (1.46)$$

и производится их умножение на \bar{w}_j^k , вычисленные в третьем слое, в виде $\bar{w}_j^k r_j^k(q^k)$.

5. В пятом скрытом слое, вычисляются оценки

$$\tilde{q}^k = \sum_{j=1}^m \bar{w}_j^k r_j^k(q^k) = \sum_{j=1}^m \frac{\prod_{l=1}^{N \times M} \mu_{jl}^k(q_l^k)}{\sum_{j=1}^m \prod_{l=1}^{N \times M} \mu_{jl}^k(q_l^k)} r_j^k(q^k), \quad (1.47)$$

6. В выходном шестом слое формируют выход \tilde{q}^k обрабатываемого изображения.

Обучение рассматриваемой нейро-нечеткой модели состоит в нахождении $(NM+1)m$ параметров b_{jl}^k моделей (1.46),

обеспечивающих наилучшее в смысле принятого критерия обучения качество обработки изображения. Для этого выражение (1.47) переписываем в виде

$$\begin{aligned}\tilde{q}^k &= \sum_{j=1}^m \bar{w}_j^k r_j^k(q^k) = \sum_{j=1}^m \bar{w}_j^k \left(b_{j0}^k + \sum_{l=1}^{N \times M} b_{jl}^k q_l^k \right) = \\ &= \bar{w}_1^k (b_{10}^k + b_{11}^k q_1^k + \dots + b_{1N \times M}^k q_{1N \times M}^k) + \\ &+ \bar{w}_2^k (b_{20}^k + b_{21}^k q_1^k + \dots + b_{2N \times M}^k q_{2N \times M}^k) + \dots + \\ &+ \bar{w}_m^k (b_{m0}^k + b_{m1}^k q_1^k + \dots + b_{mN \times M}^k q_{mN \times M}^k).\end{aligned}$$

Вводим векторы преобразования

$$r^k(q^k) = (\bar{w}_1^k, \bar{w}_1^k q_1^k, \dots, \bar{w}_1^k q_{1N \times M}^k, \bar{w}_2^k, \bar{w}_2^k q_1^k, \dots, \bar{w}_2^k q_{2N \times M}^k, \bar{w}_m^k, \bar{w}_m^k q_1^k, \dots, \bar{w}_m^k q_{mN \times M}^k)^T.$$

Вводим векторы оцениваемых параметров

$$b^k = (b_{10}^k, b_{11}^k, \dots, b_{1N \times M}^k, b_{20}^k, b_{21}^k, \dots, b_{2N \times M}^k, \dots, b_{m0}^k, b_{m1}^k, \dots, b_{mN \times M}^k) \text{ размерности } (N \times M + 1)m$$

Тогда обобщенную модель можно записать в векторной форме

$$\tilde{q}^k = b^{kT} r^k(q^k)$$

или в матричной форме

$$\tilde{Q} = BR.$$

Тогда задача в матричной форме ставится следующим образом: найти вектор \mathbf{B} , чтобы [40, 41]:

$$E = (Q - \tilde{Q})^T (Q - \tilde{Q}) \rightarrow \min,$$

где $\tilde{Q} = BR$.

Минимальное значение квадратичной невязки E достигается при $\tilde{Q} = Q$, что соответствует решению уравнения

$$Q = BR. \quad (1.48)$$

Для реальных задач количество настраиваемых параметров меньше объема выборки данных, поэтому уравнение (1.48) не имеет точного решения. В этом случае решение можно найти, используя псевдоинверсию матрицы R :

$$B = (R^T R)^{-1} R^T Q. \quad (1.49)$$

Проблемы нахождения решения (1.49) связаны с возможной сингулярностью матрицы $(R^T R)$. В связи с этим в работе предлагается нечеткий подход.

1. Пусть b^i - треугольные нечеткие число, $i = \overline{1 - (N \times M + 1)m}$:

$$\mu(b^i) = \begin{cases} \frac{r - \beta_i}{c_i} & r > \beta_i, \\ \frac{\beta_i - r}{c_i} & r \leq \beta_i. \end{cases}$$

Нечеткое число b^i в параметрическом виде можно записать в следующем виде:

$$b^i = (\beta_i, c_i).$$

Здесь β_i – центр интервала, c_i – ширина интервала $c_i \geq 0$.

Тогда решение задачи(1.49) можно выразить следующим образом:

Необходимо найти параметры (β_i, c_i) нечетких коэффициентов, которые удовлетворяли условиям:

- а) значения q_k принадлежат интервалу оценки;
- б) сумма интервалов должна быть минимальной.

Эти требования могут быть решены с помощью следующей задачи линейного программирования:

$$\begin{cases} \sum_k \sum_j c_j |r_{kj}| \rightarrow \min, \\ \sum_i \beta_i r_{ki} - \sum_i c_i |r_{ki}| \subset K \end{cases}$$

ИЛИ

$$\begin{cases} \sum_k \sum_j c_j |r_{kj}| \rightarrow \min, \\ \sum_i \beta_i r_{ki} - \sum_i c_i |r_{ki}| \leq q_k, \\ \sum_i \beta_i r_{ki} + \sum_i c_i |r_{ki}| \geq q_k. \end{cases} \quad (1.50)$$

2. Пусть b^i - Гауссовы нечеткие число, $i = \overline{1 - (N \times M + 1)m}$:

$$\mu_G(r) = e^{-\frac{1}{2} \frac{(r - \beta)^2}{c^2}}.$$

Нечеткое число b^i в параметрическом виде можно записать в следующем виде:

$$\gamma^i = (\beta_i, c_i).$$

Здесь β_i – центр интервала, c_i – ширина интервала α_s -уровня $c_i \geq 0$.

Задача ставится следующим образом:

Найти такие коэффициенты b^i с параметрами параметра (β_i, c_i) , что

а) значение q_k принадлежит множеству оценок не ниже уровня α_s ;

б) Пусть интервал оценки α_s -уровня минимален.

Интервал оценки α_s -уровня находится следующим образом:

$$d_{\alpha_s} = q_2 - q_1 = 2(q_2 - \beta).$$

Здесь $q_2 - \beta$ находится из следующего условия:

$$\alpha_s = \exp\left\{-\frac{1}{2} \frac{(q_2 - \beta)^2}{c^2}\right\} \Rightarrow d_{\alpha_s} = 2(q_2 - \beta) = 2c\sqrt{-2\ln \alpha_s}.$$

Для k -ой строки:

$$d_{\alpha_s}^k = 2c_k\sqrt{-2\ln \alpha_s}.$$

В этом случае целевую функцию можно представить в следующем виде:

$$\min \sum_{k=1}^m d_{\alpha_s}^k = \min \sum_{k=1}^m 2c_k\sqrt{-2\ln \alpha_s} = 2\sqrt{-2\ln \alpha_s} \sum_{k=1}^m \sum_{i=1}^n c_i |r_{ik}|.$$

По условию а):

$$\mu(q_k) \geq \alpha_s \Rightarrow \exp\left\{-\frac{1}{2} \frac{(q_k - \beta_k)^2}{c_k^2}\right\} \geq \alpha_s \Rightarrow \begin{cases} \beta_k + c_k\sqrt{-2\ln \alpha_s} \geq q_k, \\ \beta_k - c_k\sqrt{-2\ln \alpha_s} \leq q_k. \end{cases}$$

$$\beta_k = \sum_{i=1}^n \beta_i r_{ki}, \quad c_k = \sum_{i=1}^n c_i |r_{ki}|.$$

Чтобы найти коэффициенты нечеткой модели, решаем следующую задачу линейного программирования:

$$\begin{cases} \sum_k \sum_j c_j |r_{kj}| \rightarrow \min, \\ \sum_i \beta_i r_{ki} + \sqrt{-2\ln \alpha_s} \sum_i c_i |r_{ki}| \leq K \end{cases}$$

ИЛИ

$$\begin{cases} \sum_k \sum_j c_j |r_{kj}| \rightarrow \min, \\ \sum_i \beta_i r_{ki} + \sqrt{-2\ln \alpha_s} \sum_i c_i |r_{ki}| \geq q_k, \\ \sum_i \beta_i r_{ki} - \sqrt{-2\ln \alpha_s} \sum_i c_i |r_{ki}| \leq q_k. \end{cases} \quad (1.51)$$

Параметрическая модель с S независимыми параметрами $\alpha_1, \alpha_2, \dots, \alpha_S$ (1.51) в или S параметрическая задача в матричном виде записывается следующим образом:

$$z = r c \rightarrow \min$$

$$r(\beta + \alpha c) \subset K \quad (1.52)$$

$$\alpha \in R^S$$

Здесь $K = \{q \mid q \in R^l, q \leq |q_k|\}$ – заданное выпуклое подмножество пространства R^l .

Задачу такого типа можно назвать задачей параметрического программирования с множественно-значными коэффициентами. Ясно, что в рамках этой задачи не имеет смысла говорить о максимизации функции цели, поскольку значения этой функции – не числа, а множество чисел. В этом случае необходимо выяснить, какое отношение предпочтения в множестве альтернатив порождает эта функция, а затем исследовать вопрос о том, какие выборы считать рациональными в смысле этого отношения предпочтения.

Следующим шагом на пути уточнения рассматриваемой модели является фаззификация, т.е. описание коэффициентов задачи в форме нечетких множеств. При этом, кроме задания множеств возможных значений параметров, в модель вводится дополнительная информация в виде функций принадлежности этих нечетких множеств. Таким образом, мы пришли к постановке задачи нечеткого параметрического программирования.

Задача (1.52) сводится к следующей задаче параметрического программирования

$$z = r c \rightarrow \min,$$

$$r(\beta + \alpha c) \leq q, \quad (1.53)$$

$$\alpha \in R^S.$$

Для нахождения альтернатив, степень недоминируемости которых не меньше α , в рассматриваемом случае достаточно решить следующую задачу математического программирования:

$$z(\alpha) \rightarrow \min,$$

$$\varphi(r, z(\alpha)) \geq \alpha,$$

$$r(\beta + \alpha c) \leq q,$$

$$\psi(\alpha) = r(\beta + \alpha c) - q \leq 0,$$

где φ - функция принадлежности $z(\alpha)$.

Решением задачи (1.53) называется явным образом заданная решающая функция

$$z_s(\alpha) = \{z = rc \mid r(\beta + \alpha c) \leq q; \beta \geq 0, c > 0\}.$$

Обозначим через O_i - множество решений α удовлетворяющих (1.53). В силу того, что значения коэффициентов заданы нечетко, O_i является нечетким множеством с функцией принадлежности $\varphi_{O_i}(\alpha)$. Таким образом,

$$O_i = \{\alpha, \varphi_{O_i}(\alpha), \alpha \in R^S\}.$$

Нечеткому решению соответствует нечеткое максимальное значение

$$\varphi_{z_i} = \sup_{\alpha \in z^{-1}} \varphi_{O_i}(\alpha).$$

Для любого допустимого вектора β, c справедливы неравенства

$$z_s(\alpha) \leq g(\alpha), \quad (1.54)$$

где

$$g(\alpha) = rc.$$

Отображение q , задаваемое (1.54), описывается выражением

$$\varphi_q(z_s(\alpha), g(\alpha)) = \begin{cases} 1, & \text{если } g(\alpha) \geq z_s(\alpha), \\ 0, & \text{если } g(\alpha) < z_s(\alpha). \end{cases} \quad (1.55)$$

В связи с тем, что $z_s(\alpha)$ – элемент нечеткого множества Z_α , $g(\alpha)$ будет принадлежать некоторому нечеткому множеству G_α , которое является образом Z_α при отображении q . Согласно [38] функция принадлежности множества Z_α и его образа G_α связаны соотношением

$$\varphi_{G_\alpha}(g(\alpha)) = \max_{z_s(\alpha)} \min\{\varphi_{Z_\alpha}(z_s(\alpha)), \varphi_q(z_s(\alpha), g(\alpha))\}. \quad (1.56)$$

Подставив из (1.53) в (1.56) $\varphi_q(z_s(\alpha), g(\alpha))$, получим

$$\varphi_{G_\alpha}(g(\alpha)) = \max_{z_s(\alpha)} \{\alpha_{Z_\alpha}(z_s(\alpha)) : z_s(\alpha) \leq g(\alpha)\}. \quad (1.57)$$

Для функции $g(\alpha)$ множество O_i является прообразом G_α . Отсюда, согласно [22, с. 58], имеем

$$\varphi_{O_i}(t) = \varphi_{G_i}(g(t)) = \varphi_{G_i}((a'_0 + t'b) + e't). \quad (1.58)$$

Подставив (1.57) в (1.58), получим

$$\varphi_{O_i}(\alpha) = \max_{z_s(\alpha)} \{\varphi_{Z_\alpha}(z_s(\alpha)) : z_s(\alpha) \leq rc\}. \quad (1.59)$$

Решением (1.59) для $\varphi_{Z_\alpha}(z_s(\alpha))$ является

$$\varphi_{z_i}(\alpha) = \exp\left(-\frac{k_i}{2}[\max(0, z_s(\alpha) - rc)]^2\right). \quad (1.60)$$

Нечеткое отображение зависит от s переменных, т.е. $\alpha = \alpha_1 \times \alpha_2 \times \dots \times \alpha_s$ — декартово произведение соответствующих множеств. В общем случае функция принадлежности этого подмножества имеет вид

$$\varphi_z(\alpha) = \prod_{i=1}^s \varphi_{z_i}(\alpha). \quad (1.61)$$

Из (1.60) и (1.61) имеем

$$\varphi_z(\alpha) = \exp(-\Phi(\alpha)), \quad (1.62)$$

где

$$\Phi(\alpha) = \frac{1}{2} \sum_{i=1}^s k_i [\max(0, z_s(\alpha) - rc)]^2.$$

Нами доказано следующее утверждение 1.

Пусть $z_s(\alpha)$ нечеткое решение задачи (1.53). Тогда множество решений имеет функцию принадлежности (1.62).

Если имеется нечеткая информация, то нужно выбрать такую оценку параметров α , чтобы она минимизировала $z_s(\alpha)$ и в то же время ее степень принадлежности допустимому множеству Z_α была максимальной, т.е. :

$$z(\alpha) \rightarrow \min, \varphi_z(\alpha) \rightarrow \max. \quad (1.63)$$

Согласно лемме 2 [23, с.61], в силу положительности $\varphi_z(\alpha)$ второй критерий в (1.63) можно заменить на $\ln \varphi_z(\alpha)$. Таким образом (1.63) эквивалентно задаче

$$z(\alpha) \rightarrow \min, \Phi(\alpha) \rightarrow \min. \quad (1.64)$$

Предпочтительными решениями такой задачи являются те, которые нельзя улучшить по одному критерию, не увеличив при этом другой критерий в (1.64).

Функция $z(\alpha)$ выпуклая функция, $\Phi(\alpha)$ выпуклая функция. В этом случае решения оптимальные по Парето являются решением задачи [23, с.60]

$$\begin{aligned} L(\alpha) = z(\alpha) + h\Phi(\alpha) &\rightarrow \min \\ h &\in [0, \infty] \end{aligned} \quad (1.65)$$

В случае формализации информации в виде нечетких множеств определяется не одно решение, а некоторое их множество. Они являются функцией h . Обозначим их через $\alpha(h)$.

В силу выпуклости функции $L(\alpha)$ для $h_2 > h_1 \geq 0$ имеем

$$\begin{aligned} z(h_1) + h_1\Phi(h_1) &< z(h_2) + h_1\Phi(h_2), \\ z(h_2) + h_2\Phi(h_2) &< z(h_1) + h_2\Phi(h_1). \end{aligned}$$

Отсюда получим

$$\begin{aligned} h_2(\Phi(h_2) - \Phi(h_1)) + h_1(\Phi(h_1) - \Phi(h_2)) &< 0, \\ (\Phi(h_2) - \Phi(h_1))(h_2 - h_1) &< 0. \end{aligned}$$

Из данного неравенства следует утверждение 2: Если $L(\alpha)$ – выпуклая функция, то $\varphi_z(\alpha)$ – строго монотонно убывает при $h \geq 0$.

Это означает, что в множестве O_i нет такой альтернативы, для которой одновременно выполнялись бы неравенства

$$\varphi_o(\alpha) > \varphi_z(h) > 0 \text{ и } z(\alpha) > h,$$

т.е. нет такого элемента α , который имел бы большую, чем $\varphi_z(h)$, степень принадлежности множеству φ_o и давал бы большее, чем h максимизируемой функции. Если лицо принимающее решение предпочитает выбрать в качестве решения конкретную альтернативу α то его выбор должен опираться не только на степень принадлежности этой альтернативы нечеткому множеству $\varphi_z(\alpha)$, но и на соответствующие значения функции $z(\alpha)$.

Тогда параметры нечеткого числа q находятся в следующем виде:

центр интервала:

$$b_q = \sum_j \beta_j r_{ij} = \beta^T r,$$

ширина интервала:

$$c_q = \sum_j c_j |r_{ij}| = c^T |r|.$$

Проведен вычислительный эксперимент. База правил количественной оценки указанных параметров для исходного изображения и полученного после обработки приведены в таблице 1. При составлении логических правил учитывалось, что качество изображения ухудшается при большом наличии шумов, плохой яркости и контрастности. При отсутствии этих факторов или их снижении качество изображения улучшается. Модель оценки качества изображения основывается на нечетком логическом выводе Сугено, в котором все значения входных и выходных переменных заданы нечеткими множествами.

Входной сигнал задается в виде матрицы, столбцы которой соответствуют разным классам обрабатываемого изображения. Как говорилось выше в нулевом слое этот сигнал преобразуется в вектор с элементами q_i^k где $i = \overline{1, N \times M}$, k принимает значения — от нуля до 3 в нашей задаче, т.е. $q_i^0 = f$, $q_i^1 = g_1$, $q_i^2 = g_2$, $q_i^3 = g_3$ соответственно.

Таблица 1.4

База правил количественной оценки

Но- мер пра- вила	<i>Если «входы»</i>				<i>То</i>
	Яркость q_1	Контра- ст q_2	Шум q_3	Смещен- ие фокуса q_4	Качество изображения r
	В	В	В	В	$r = 1,6 + 4,7 q_1 - 66,9 q_2 + 37,4 q_3 + 2,8 q_4$
	В	В	В	Г	$r = 1,6 + 30,4 q_1 - 71,7 q_2 + 52,1 q_3 - 0,7 q_4$
	В	В	В	Е	$r = 2,0 + 392,5 q_1 - 354,5 q_2 + 12,8 q_3 - 4,7 q_4$
	В	В	Г	В	$r = 0,3 + 0,1 q_1 + 50,1 q_2 + 3,0 q_3 - 4,7 q_4$
	В	В	Г	Г	$r = -3,2 - 13,5 q_1 + 35,4 q_2 + 9,9 q_3 - 2,0 q_4$
	В	В	Г	Е	$r = 1,2 - 3,1 q_1 + 1,1 q_3 - 0,6 q_4$
	В	В	Е	В	$r = 2,1 + 28,7 q_1 + 56,1 q_2 - 0,1 q_3 - 8,7 q_4$
	В	В	Е	Г	$r = -0,7 + 64,0 q_1 + 31,1 q_2 + 2,7 q_3 - 9,5 q_4$
	В	В	Е	Е	$r = 5,5 + 6,4 q_1 - 10,6 q_2 - 3,4 q_3 + 0,7 q_4$

Функции принадлежности нечетких термов, используемых в этой базе знаний, выбирались экспертно (рис.1.38). Нейро-нечеткая сеть выбранная этими функциями принадлежности обеспечивает грубую аппроксимацию объекта.

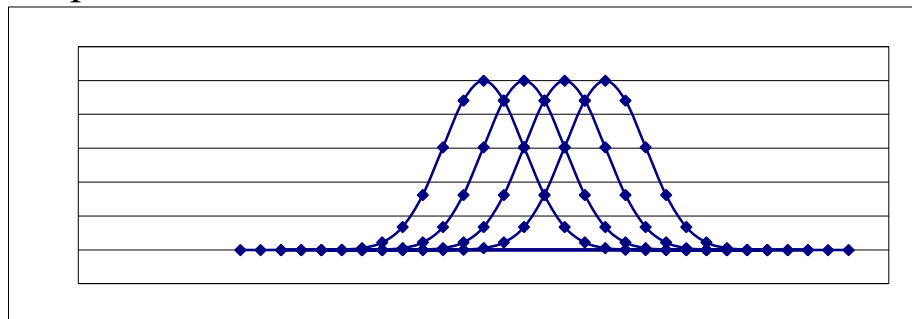


Рис.1.38. Функция принадлежности выходных термов до обучения

Функции принадлежности нечетких термов после обучения приведены на рис. 1.39.

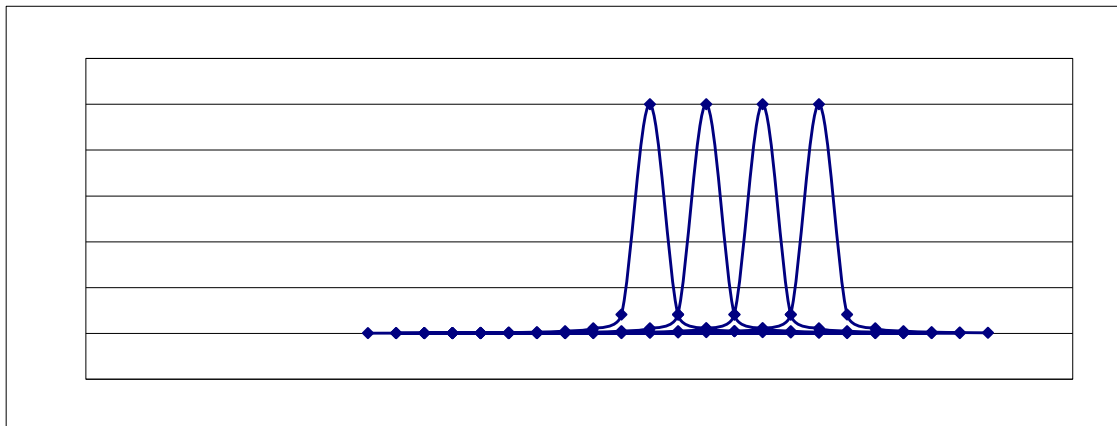


Рис.1.39. Функции принадлежности входных термов после обучения

Из рисунков 1.38, 1.39 и табл. 1.5 можно видеть, что в результате обучения нейро-нечеткой сети наибольшим изменениям подверглись параметры с функцией принадлежности

$$\mu_G(r) = e^{-\frac{1}{2} \frac{(r-\beta)^2}{c^2}},$$

тогда как остальные параметры практически не изменились. Выбор экспертом больших значений параметров c свидетельствует о значительной неопределенности в оценке нечетких термов. Уменьшение значений параметров c в ходе обучения привело к «концентрации» функций принадлежности, что свидетельствует о снятии неопределенности в оценки нечетких термов.

Фрагмент расчетов нейро-нечеткой аппроксимации

$\frac{\partial \mu}{\partial c}$	$\frac{\partial \mu}{\partial \beta}$	$\frac{\partial E}{\partial c}$	$\frac{\partial E}{\partial \beta}$	c	β
До обучения					
				1	0,32
				1	0,30
				1	0,29
				1	0,27
				1	0,26
				1	0,25
				1	0,23
				1	0,22
				1	0,21
				1	0,19
После обучения					
0,21	1,20	-8,12	-41,86	0,29	0,30
0,11	0,93	-4,6	-32,47	0,29	0,28
0,04	0,60	-1,97	-10,28	0,29	0,26
0,22	0,28	3,27	4,10	0,28	0,25
0	0	0	0	0,29	0,23
0,02	-0,28	0,17	-4,10	0,28	0,22
0,03	-0,50	0,68	-8,86	0,28	0,21
0,09	-0,81	1,79	-5,10	0,28	0,20
0,17	-1,07	3,17	-5,10	0,28	0,18
0,27	-1,30	4,84	-5,09	0,28	0,24

Методы нечеткой логики являются мощным инструментом для представления и обработки знаний в виде правил вида „если—то“, системы нечеткого вывода разрабатываются быстрее и получаются более простыми по сравнению с четкими аналогами

В настоящее время математическая морфология широко используется при цифровой обработке изображений. К основным морфологическим операциям относятся операции расширения серого цвета и эрозии:

$$(f \oplus b)(s,t) = \max_{\substack{(s-x,t-y) \in D_f \\ (x,y) \in D_b}} (f(s-x,t-y) + b(x,y)); \quad (1.66)$$

$$(f \ominus b)(s,t) = \min_{\substack{(s+x,t+y) \in D_f \\ (x,y) \in D_b}} (f(s+x,t+y) - b(x,y)); \quad (1.67)$$

где: $f(s,t)$ – функция яркости изображения;

$b(x,y)$ – структурный элемент;

D_f, D_b – зоны обнаружения f и g .

Предположим, что обрабатываемое изображение задается функцией яркости $f: P \rightarrow [0,1]$ в пиксельной плоскости $P = \{(x,y) | x \in \{1,2,\dots,N\}, y \in \{1,2,\dots,M\}\}$. В этом случае нетрудно увидеть, что функция яркости изображения после применения операции (1.66) или (1.67) выходит за пределы диапазона $[0,1]$, что не позволяет использовать дилатацию или эрозию в качестве самостоятельной операции с изображениями.

Для преодоления выявленных недостатков морфологических операций нам было предложено использовать теорию неясных множеств.

Мы определяем серые изображения и элементы, составляющие структуру, с неясными подмножествами следующего вида:

$$\underline{A} = \{p | \mu_{\underline{A}}(p), p \in P\}$$

Функция релевантности изображения определяется его функцией яркости:

$\mu_{\underline{A}}(x) = f_{\underline{A}}(x)$, дана априорная функция структурного элемента. В этом случае, согласно предлагаемому подходу, дилатация может быть обобщена до неопределенного состояния следующим образом:

$$\underline{A} \oplus \underline{B} = \left\{ c \mid \exists x: (x \in (\hat{B})_c) \wedge (x \in \underline{A}) \right\} = \left\{ c \mid \max_{x \in P} \left(\mu_{(\hat{B})_c}(x) \wedge \mu_{\underline{A}}(x) \right) \right\}.$$

Предлагается использовать следующее выражение для обобщения операции эрозии на состояние изображения:

$$\underline{A} \ominus \underline{B} = \left\{ c \mid \min_{x \in P} \left(\min \{1, 1 - \mu_{(\hat{B})_c}(x) + \mu_{\underline{A}}(x)\} \right) \right\}.$$

Морфологический метод выделения границ, сформированных на основе операций дилатации и эрозии, до нечеткого состояния выглядит так:

$$\underline{A}' = (\underline{A} \oplus \underline{B}) \setminus (\underline{A} \ominus \underline{B}),$$

где $\mu_{\underline{A} \setminus \underline{B}}(x) = \max\{\mu_{\underline{A}}(x) - \mu_{\underline{B}}(x), 0\}$.

Предлагаются алгоритмы обнаружения глазных болезней с помощью нейронных сетей. Использовалась свёрточная нейронная сеть, к которой применялся алгоритм обучения методом обратного распространения ошибки.

Свёрточная нейронная сеть состоит из разных слоев:

Первые два типа слоев (свёрточный, субдискретизирующий) чередуются, чтобы сформировать нежелательный вектор символов для многослойного перцептрона.

Размеры всех карт свёрточного слоя рассчитываются по следующей формуле:

$$(w, h) = (mW - kW + 1, mH - kH + 1),$$

где (w, h) – рассчитанный размер карты свёрточного слоя; mW – ширина предыдущей карты; mH – высота предыдущей карты; kW – ширина ядра; kH – высота ядра.

Ядро состоит из фильтра или окна, которое скользит по всей области предыдущей карты и находит определенные отметки объектов. Ядро перемещается по предыдущей карте и выполняет операцию свёртки в соответствии со следующей формулой, которая часто используется при обработке изображений.

$$(f * g)[m, n] = \sum_{k, l} f[m - k, n - l] * g[k, l],$$

где f – заданная матрица изображения; g – ядро свёртки.

В процессе сканирования карты предыдущего слоя с ядром подслоя (фильтром) ядро сканирования не пересекается, в отличие от свёрточного слоя. Обычно каждая карта имеет ядро с размером 2x2, что позволяет уменьшить предыдущие карты свёрточного слоя в 2 раза. Все карты признаков разбиты на ячейки по 2x2 элемента, из которых выбирается максимальное значение.

Обычно функция активации ReLU используется на уровне подвыбора.

Слой можно выразить следующей формулой:

$$x^l = f(a^l * \text{subsample}(x^{l-1}) + b^l),$$

где x^l - выход слоя; $f()$ - функция активации; a^l, b^l - коэффициенты сдвига слоя; `Subsample()` - операция выбора локальных максимальных значений.

Чтобы уменьшить квадратичную ошибку обучения нейронной сети, мы используем метод градиентного спуска, вычисляя произведение частоты для каждого веса. Получаем следующие отношения:

$$\frac{\partial E_n^p}{\partial w_i} = x_{n-1}^j \cdot \frac{\partial E_n^p}{\partial y_n^i},$$

$$\frac{\partial E_n^p}{\partial y_n^i} = g'(x_n^j) \cdot \frac{\partial E_n^p}{\partial x_n^i},$$

где g' - активация функции

$$\frac{\partial E_n^p}{\partial y_n^i} = x_n^i - d_n^i,$$

$x_{n-1}^j - j$ - выход слоя $(n-1)$ - выход нейрона, $y_n^i - (n-1)$ - скалярное произведение всех выходов нейронов слоя и соответствующих весовых коэффициентов.

Параметры глазных болезней были смоделированы на основе характеристик сосудов при обнаружении глазных болезней с использованием анализов и экспериментов:

- расчет толщины сосуда $a = \frac{\sum_{i=1}^K d_i}{K}$ $v = \frac{\sum_{i=1}^K D_i}{K}$ и при наличии $v > \frac{2}{3}a$

ставится диагноз глазных болезней (рисунок. 1.40 а) где d_i - диаметр круга, втянутого в артерию, D_i - диаметр круга, втянутого в вену, K - количество кругов.

Угол, под которым образуются стороны А и В, вычисляется через $tg\alpha$ равный сторонам образовавшегося треугольника.

$$\alpha = \arctg \frac{c_1}{b} + \arctg \frac{c_2}{a}$$

угол рассчитывается через пиксели, и проводится диагностика для состояния тупого угла. (рисунок 1.40 б).

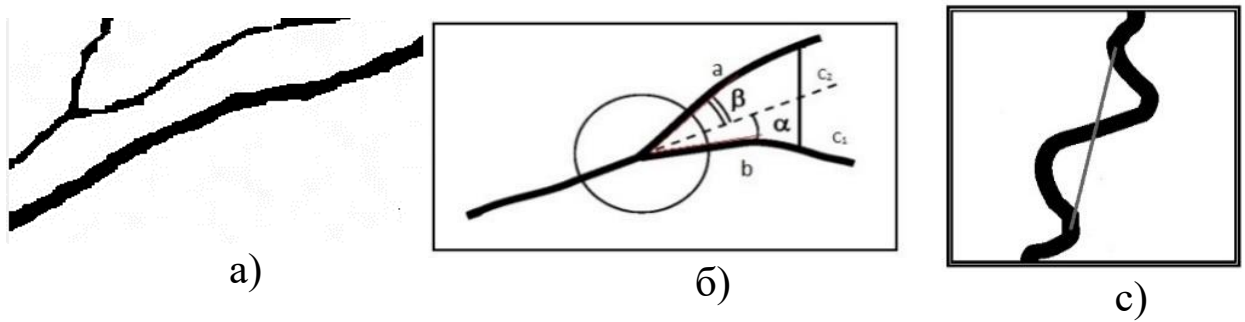


Рис. 1.40. Формы видакровоносных сосудов глазного дна

Если функция задана в графической форме или в виде сложного аналитического выражения, возникают другие типы вариантных проблем. Например, пространство $[a, b]$ следует разделить на l частей:

$$[a_{i-1}, a_i], i = 1, \dots, l, a_0 = a, a_l = b,$$

Функция каждого отрезка $[a_{i-1}, a_i]$ должна быть приближена к следующему многочлену

$$f(x) \approx g(x) = \sum_{k=0}^n c_k x^k$$

Среди таких способов конвергенции в том или ином смысле найдется альтернатива. Часто этому предшествует требование $n_1 = \dots = n_l, m_1 = \dots = m_l, l$ - это количество делений, которые делятся, а метод чередуется по a_1, \dots, a_{l-1}, n и m .

В частности, то есть в сечении, заданном при $l = 1$, возникает проблема плоского приближения к многочлену (рисунок. 1.40 с). На основе изображения сосудов глазного дна, вид спереди, была разработана функциональная структура программы (рис. 1.41).

После того, как вы создали и архитектуру свёрточной нейронной сети и обучили ее, вы можете передать ее на использование. Построение сверточной нейронной сети, классификатора с использованием правильно распределенных нейронных сетей начинается с первого сверточного слоя.



Рис. 1.41. Функциональная структура программы

Архитектура сформированной свёрточной нейронной сети представлена на рисунке 1.42.

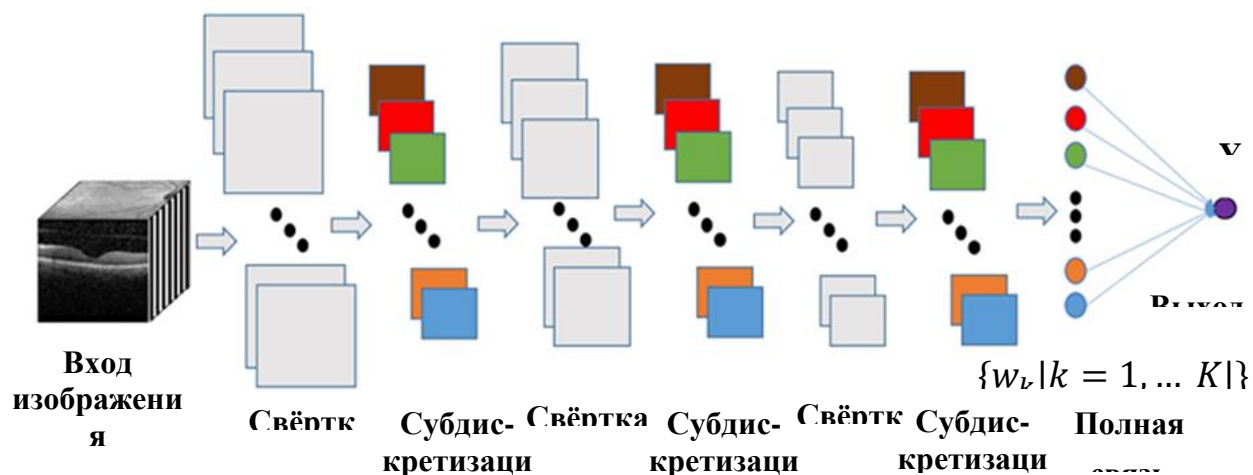


Рис. 1.42. Общая структура свёрточной нейронной сети

Свёрточная нейронная сеть, показанная на рисунке 4, состоит из следующих типов слоев: свёрточные слои (convolutional), субдискретизирующие слои (слои субдискретизации, выборки). При обертывании используется только небольшая матрица с ограниченным весом или зеркало размером 3×3 .

После первого слоя получаем матрицу $28 \times 28 \times 1$ - функцию активации или карту символов, т.е. 784 значения. Тогда S1 матрица, полученная в слое, подвергается операции субдискретизации (сжатию) с использованием окна размера $k \times k$. Ниже разработан гибридный алгоритм, основанный на боковом изображении сосудов глаза.

На основе бокового изображения сосудов глаза была разработана функциональная структура программы (рис. 1.43).

На основе предложенных интеллектуальных гибридных методов и алгоритмов разработана программа диагностики глазных болезней по изображениям глазного дна и анализа результатов.

Программное обеспечение было разработано на Python версии 3.6.7. Для работы программы требуется видеокарта. При обработке изображений загружаются следующие библиотеки:

absl-py==0.10.0 astunparse==1.6.3 attrs==20.2.0
cachetools==4.1.1 certifi==2020.6.20 chardet==3.0.4 cycler==0.10.0
dill==0.3.2 future==0.18.2 gast==0.3.3 google-auth==1.21.1 google-
auth-oauthlib==0.4.1 google-pasta==0.2.0 googleapis-common-
protos==1.52.0 grpcio==1.32.0 h5py==2.10.0 idna==2.10 importlib-
metadata==1.7.0 Keras==2.4.3 Keras-Preprocessing==1.1.2
kiwisolver==1.2.0 Markdown==3.2.2 matplotlib==3.3.1
numpy==1.18.5 oauthlib==3.1.0 opencv-python==3.4.3.18 opt-
einsum==3.3.0 Pillow==7.2.0 promise==2.3 protobuf==3.13.0
yasn1==0.4.8 pyasn1-modules==0.2.8 pyparsing==2.4.7 PyQt5==5.15.1
PyQt5-sip==12.8.1 python-dateutil==2.8.1 PyYAML==5.3.1
requests==2.24.0 requests-oauthlib==1.3.0 rsa==4.6
scipy==1.4.1 six==1.15.0 tensorboard==2.3.0 tensorboard-plugin-
wit==1.7.0 tensorflow==2.3.0 tensorflow-datasets==3.2.1 tensorflow-
estimator==2.3.0 tensorflow-metadata==0.24.0 termcolor==1.1.0
tqdm==4.48.2 urllib3==1.25.10 Werkzeug==1.0.1 wrapt==1.12.1
zipp==3.1.0.

Экспериментальные исследования проводились на изолированных тестовых изображениях при сосудистой патологии.

Набор данных бокового обзора сосудов глаза был взят с открытого сайта набора данных kaggle.com. Изображения из этого набора данных были распределены следующим образом. Таблица 2.

На основе этого набора данных были получены результаты анализа различных структур.

В ходе исследования было создано программное обеспечение для распознавания и классификации объектов на основе метода обучения с учителем, обернутой нейронной сети с одним слоем входа, тремя свёрточными слоями и тремя субдискретизирующими слоями, созданными в работе.

Результаты, основанные на структуре сети, представлены в таблице 1.6.

Оптимальное количество эпох (итераций) определялось путем обучения, модификации и тестирования выбранной сети.

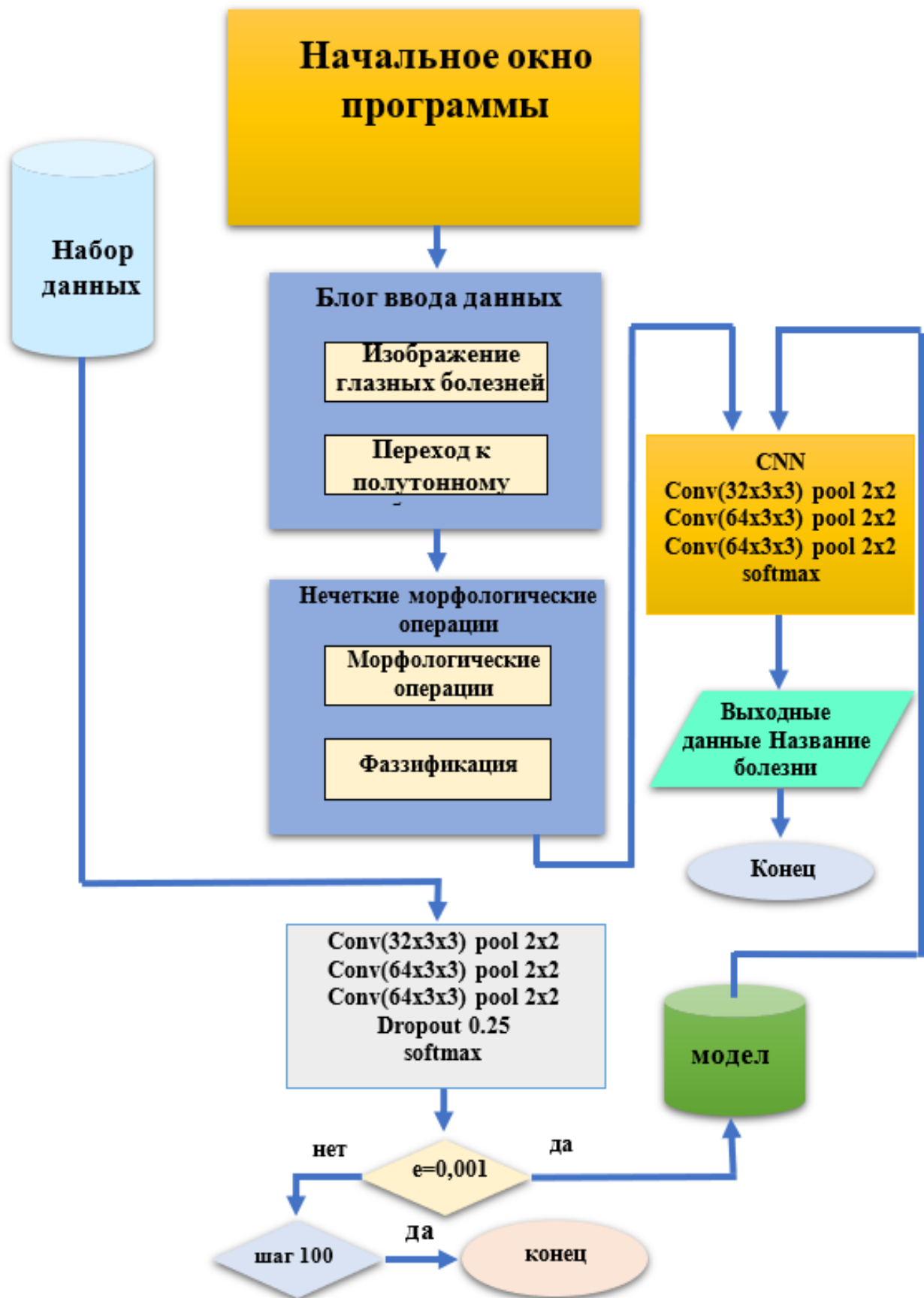


Рис. 1.43. Функциональная структура программы на основе бокового изображения сосудов глаза

Таблица 1.6

Распределение изображений, использованных в
экспериментах

	Здоровый глаз	Новый кровеносный сосуд, растущий из хориоида	Друзовская болезнь глаз
Общее количество изображений	26315	37205	28616
Количество обученных изображений	242	242	242
Количество протестированных изображений	26557	37447	28858

Таблица 1.7

Незнакомые системе тестовые данные являются результатом
внедрения

№	Название категории	Всего тестов	Верно	Неправильный	Процент
1	Новый кровеносный сосуд, растущий из хориоида	242	212	30	87,6
2	Друзовская болезнь глаз	242	199	43	82,3
3	Здоровый глаз	242	232	10	95,8

На выходе свёрточной нейронной сети получаются вероятности совпадения объекта, присутствующего в базе данных. Кадр с максимальной вероятностью выбирается и принимается как окончательный результат на тот момент. Количество ошибок составляло 4-40%.

Результаты идентификации в таблице 1.8 были получены при проведении морфологической фильтрации.

Таблица 1.8.

Результаты, когда тестовые данные, незнакомые системе, передаются через морфологическую фильтрацию

№	Название категории	Всего тестов	Верно	Неправильный	Процент
1	Новый кровеносный сосуд, растущий из хориоида	242	208	34	85,9
2	Друзовская болезнь глаз	242	192	50	79,3
3	Здоровый глаз	242	221	21	91,3

Результаты таблицы 1.8 были получены после проведения морфологической фильтрации. Количество ошибок 9-36%.

Результаты идентификации в таблице 1.9 были получены при выполнении нелинейной морфологической фильтрации.

Таблица 1.9

Когда тестовые данные, незнакомые системе, препедаются через нечеткую морфологическую фильтрацию

№	Категория номи	Всего тестов	Верно	Неправильный	Процент
1	Новый кровеносный сосуд, растущий из хориоида	242	229	13	94,6
2	Друзовская болезнь глаз	242	218	24	90,2
3	Здоровый глаз	242	235	7	97,4

Результаты таблицы 1.9 были получены после нечеткой морфологической фильтрации. Количество ошибок составляло 5-7%. Точность расчетов повысилась при проведении нечеткой морфологической фильтрации.

Сверточные нейронные сети обеспечивают частичную устойчивость против изменения масштабов, сдвигов, скручиваний, смены ракурса и других искажений.

Разработанная автоматизированная система анализа изображений глазного дна на основе методов и алгоритмов количественной оценки патологических изменений сосудистой системы глазного дна в организме позволяет диагностировать другие заболевания сетчатки.

В данном исследовании автоматизированная система анализа изображений глазного дна позволяет автоматизировать отдельные этапы диагностики.

Анализ литературных источников по применению нечетких методов в задачах обработки цифровых изображений показывает, что эти методы используются в основном в следующих задачах: повышение контрастов изображений, выделение контуров и сегментация. Однако, эти методы не нашли широкого распространения при решении прикладных задач обработки изображений из-за их вычислительной сложности. Учитывая это, в работе определена цель исследования и сформулированы задачи обработки изображений с использованием аппарата нечетких множеств. Определены состав и структура системы обработки изображений, основанной на теории нечетких множеств. Разработана структурная схема основных функциональных подсистем с учетом применения «мягких вычислений». Предложена схема предварительной обработки и оценки качества изображений с использованием аппарата нечетких множеств. Проработаны вопросы формирования баз правил на модельном примере. Путем компьютерного эксперимента показано, что применение предложенной нейро-нечеткой сети существенно снижает затраты на обучение. Для нейронной сети число итераций обучения составляет 8400, а время 7 мин; для нейро-нечеткой сети число итераций составляет 1150, а время 1,2 мин.

Глава 2. ПРИМЕНЕНИЕ СОВРЕМЕННЫХ НЕЙРОСЕТЕЙ ПРИ СОЗДАНИИ ИЗОБРАЖЕНИЙ

2.1. Как рисовать с помощью нейросетей

Рисование в нейросетях сегодня набирает все большую популярность и количество людей, которые следят за развитием искусственного интеллекта неуклонно растет.

О Midjourney говорит весь интернет: искусство, которое творит эта нейросеть, легко спутать с произведениями профессиональных художников. При этом стоит труд искусственного интеллекта в тысячи раз дешевле, а время создания картин измеряется секундами.

Как устроена эта дьявольская машинка изнутри, кто ее придумал и зачем миру живые дизайнеры и художники, если есть Midjourney?

Midjourney

Midjourney — разработанная одноименной американской компанией нейросеть, которая поразила пользователей по всему миру картинками (иначе назвать это невозможно), созданными на основе текстовых запросов. В феврале 2022 года проект основал ученый и предприниматель Дэвид Хольц — 33-летний выпускник факультета прикладной математики Университета Северной Каролины в Чапел-Хилл. В студенчестве Хольц успел поработать в Институте Макса Планка, где изучал алгоритмы нейровизуализации и разрабатывал карту мозга крысы на клеточном уровне, и даже в исследовательском центре NASA, где занимался технологией LiDAR (обнаружение и определение дальности с помощью света, используется, в том числе, в камерах iPhone для лучшей фокусировки на объекте).

В 2011 году молодой ученый ушел из аспирантуры и переехал в Сан-Франциско, где основал первую собственную компанию — LeapMotion (сейчас Ultraleap), которая разрабатывает датчики движения и другие системы распознавания человеческих жестов. Хольц руководил фирмой вплоть до 2021 года, но решил выйти из бизнеса: ему, по собственным словам, не хотелось управлять крупной компанией, — в интервью TheVerge Хольц признался, что ему интересна другая, молодая и быстроразвивающаяся среда.

В начале 2022 года Хольц вышел из состава учредителей Ultraleap и основал Midjourney. По словам предпринимателя, штат компании даже сейчас, когда ее настигла мировая слава, не превышает 10 человек, у проекта нет инвесторов, а деньги не являются главной мотивацией основателя. «Главное, чего я хочу, — чтобы на ближайшие 10 лет у нас был дом, в котором мы могли бы экспериментировать с технологиями и создавать продукты, которые будут иметь значение не только для меня, но и для всего мира. Ну и получать удовольствие в процессе», — говорит он.

Как это работает

Работу Midjourney обеспечивают два технологических прорыва в области искусственного интеллекта, произошедших относительно недавно: умение нейросетей понимать человеческую речь и создавать образы. Чтобы преобразовать два этих скилла в стройную систему, которая по запросу выдает произведения искусства, нейронную сеть обучают выстраивать соответствие между текстовыми описаниями и визуальными образами на сотнях миллионов примеров. Результаты такого обучения позволяют решать различные кросс-модальные задачи — генерацию картинок по текстовому описанию, генерацию текстовых описаний по картинкам, дорисовку частей изображения, и так далее, говорит руководитель управления экспериментальных систем машинного обучения SberDevices Сергей Марков. «Midjourney — это диффузионная нейронная сеть и состоит как бы из двух нейросетей: первая отвечает за обработку и понимание текста, вторая — за генерацию изображения», — объясняет Марков.

В середине июля Midjourney вышла в фазу бета-тестирования и стала доступна пользователям по всему миру. Правда, чтобы дать задание Midjourney, нужно быть зарегистрированным в Discord — кросс-платформенном мессенджере, популярном у геймеров, разработчиков игр и дизайнеров. Сначала нужно зайти на официальный сайт Midjourney и авторизоваться через Discord, затем оплатить подписку или воспользоваться бесплатной версией. Бесплатная версия позволяет сгенерировать и скачать 12 изображений, но не дает доступ к личному кабинету (это мешает отслеживать судьбу своих запросов в общем чате), за \$10 можно

создавать до 200 изображений в месяц, за \$30 — генерировать бесконечное количество картинок. Доступна также корпоративная подписка стоимостью \$600, которая дает сотрудникам компании возможность создавать картинку в команде и просматривать индивидуальные работы друг друга.

По словам Хольца, способ доступа к системе через Discord он выбрал из-за группового принципа работы платформы: люди охотнее фантазируют, когда собираются в группы, считает Хольц. Присоединившись к сервису, можно отправлять текстовые команды для создания изображений вместе с другими пользователями или поодиночке на любом из множества каналов Discord.

Чтобы создать изображение, достаточно внести в чат с ботом Midjourney слова, описывающие картину, которую в итоге хочется получить. Система сгенерирует четыре изображения на выбор, а дальше наиболее подходящую картинку можно масштабировать, изменять и дорабатывать до идеала.

Полученные изображения появляются в общем канале Discord примерно через минуту после отправки запроса. Обладатели платной подписки могут отправлять боту команды в формате личных сообщений, а не через общедоступный канал. Но сгенерированные нейросетью изображения по умолчанию остаются общедоступными для просмотра.

Ничего оригинального

Midjourney не первая и не единственная нейросеть, способная генерировать изображения на основе текстовых запросов, — до этого мир потрясли DALL-E от OpenAI и Imagen от Google.

Разработку DALL-E обеспечил грант в \$1 млрд от Microsoft, который корпорация выделила проекту OpenAI, специализирующемуся на технологиях искусственного интеллекта. На эти средства OpenAI разработал нейросеть на базе третьего поколения алгоритма обработки естественного языка GPT-3, который компания создала годом ранее. Для широкой аудитории нейросеть стала доступна через два года — в январе 2021-го. А в апреле 2022 года вышла ее улучшенная версия — DALL-E 2.

В ноябре 2021 года команда разработчиков Sber AI создала аналогичную нейросеть, которая обрабатывает запросы на русском языке. Проект получил название ruDALL-E, но отношения

к оригинальному DALL-E не имеет: разработки засекречены, и российские специалисты самостоятельно обучали нейросеть с нуля. Разработчики заявляют, что процесс обучения ruDALL-E стал самой большой вычислительной задачей в России.

В мае 2022 года нейросеть Imagen, генерирующую изображения, выпустил Google. На официальном сайте продукта разработчик утверждает, что Imagen превосходит DALL-E по точности изображений и соответствию картинки техническому заданию.

Никаких технологических «киллер-фичей» у Midjourney нет, уверен директор по науке и технологиям консалтинговой компании «Агентство Искусственного Интеллекта» Роман Душкин. По его словам, все перечисленные системы работают по общему принципу: человек дает как можно более конкретное текстовое описание той картинке, которую он хочет получить. Благодаря технологии глубокого обучения, нейросеть понимает смысл текста и преобразует его в картинку. Общая модель работы нейросетей такого типа открыта и понятна: о ней пишут подробные статьи в научных журналах. Разница и конкурентное преимущество — в индивидуальных технологических особенностях, которые влияют на пользовательский опыт и которые разработчики не раскрывают. Например, Midjourney обеспечивает более высокое разрешение итоговых картинок и большую детализацию в соответствии с текстовым запросом, отмечает руководитель лаборатории искусственного интеллекта в новых материалах центра в сфере ИИ на базе университета «Иннополис» Руслан Лукин.

Но все же главный «изюм» Midjourney в том, что ею может воспользоваться любой желающий, в отличие от других подобных нейросетей, доступ к которым возможен только по приглашениям или платно, считает исследователь разработчика ИИ-продуктов DeepPavlov.ai Александр Чернявский. «Среди всех генерирующих картинки из текста нейросетей это первая широкодоступная и частично бесплатная сеть, которая благодаря этой особенности „завирусилась“ в интернете», — говорит он. DALL-E 2 тоже можно использовать свободно, но доступ к ней открыли совсем недавно, а Midjourney изначально пошла по пути массового использования и не прогадала, добавляет Сергей Марков из Sber AI:

«Пути хайпа не всегда поддаются рациональному анализу, но, безусловно, Midjourney сегодня — это одна из лучших моделей для создания изображений по текстовым описаниям среди доступных широкой публике».

По его прогнозу, несмотря на преимущества, хайп вокруг Midjourney продлится недолго: через какое-то время ей на смену придут еще более совершенные модели, так как прогресс в этой области сейчас очень быстрый, считает эксперт.

Как на этом заработать

Зачем нам нужна Midjourney и ее аналоги? Исключительно для развлечения или для чего-то еще?

По мнению Романа Душкина из «Агентства Искусственного Интеллекта», у функции визуализации нейросетей большой спектр практического применения, начиная от маркетплейсов, где нужно генерировать множество однотипных картинок для представления товаров на сайте, до создания художественных произведений и генерации NFT. Руководитель группы исследований компании по разработке технологий искусственного интеллекта Just AI Мурат Апишев к этому списку добавляет генерацию маркетингового и внутриигрового контента, дизайна, анимации и даже кино. Сергей Марков убежден, что подобные Midjourney системы найдут применение в интерьерном дизайне и дизайне упаковок, генерации прототипов веб-страниц, элементов интерфейсов, эмодзи и стикеров, в гейм-индустрии для создания персонажей или целых миров, в архитектуре, разработке логотипов и фирменного стиля, дизайне одежды, боди-арте, ивент-индустрии, иммерсивных театрах — «и это еще далеко не полный список». Монетизировать такие системы можно множеством различных способов, рассуждает Душкин, но «намного более перспективным, чем подписка, кажется создание крупных шедевров, которые можно продавать на аукционах».

Инструмент или конкурент?

В июне 2021 года журнал Cosmopolitan вышел с первой в мире обложкой, созданной нейросетью. На картинке изображена женщина-космонавт, будто бы снятая на широкоугольный объектив, а надпись на ней гласит: «это заняло всего 20 секунд». Автором обложки была нейросеть DALL-E.

Эта история вызвала бурные обсуждения в Twitter: часть пользователей восхищались работой нейросети, часть — хейтили ее, а кто-то всерьез обеспокоился тем, что машины в ближайшем будущем отберут работу у художников и дизайнеров, ведь шедевры уровня именитых мастеров теперь можно создавать за пару минут и пару десятков долларов в месяц.

Основательница арт-компании Everal Люси Виноград призывает по этому поводу не беспокоиться: по ее словам, нейросети были и остаются только инструментом для визуализации, но не полноценным творцом. «Искусство — это в первую очередь идея, а нейросети генерируют лишь бездумную красоту. Без человека они беспомощны», — уверена предпринимательница. При этом искусственный интеллект уже решает многие задачи визуальной составляющей дизайна, и здесь им правда нет равных, добавляет она.

Художник и сооснователь сети квестов «Клаустрофобия» Богдан Кравцов считает нейросеть мощным инструментом в руках художника, «максимум — его соавтором», но точно не полноценной заменой человека. «Именно живой человек ставит задачу, настраивает параметры нейросети, а в конце работы — фильтрует результаты, отбирая удавшиеся дубли в соответствии с собственным эстетическим чувством», — говорит Кравцов.

Но самое главное, чего не хватает нейросети, чтобы стать художником, — способности захотеть создать картину».

Отбор результатов, разделение вариантов на более и менее удачные — действительно важная задача, которая пока не под силу искусственному интеллекту, согласен Роман Душкин из «Агентства Искусственного Интеллекта»: «Эмоциональный отклик — это то, ради чего мы воспринимаем искусство. А оценить будущий эмоциональный отклик, который вызовет произведение у человека, может только человек, ведь у нейросетей эмоций нет, — напоминает он. — Поэтому созданные ИИ шедевры — это всегда человеко-машинное творчество и не стоит бояться использовать этот мощный инструмент для создания шедевров».

Сам основатель Midjourney Дэвид Хольц не планирует отбирать работу у живых художников и дизайнеров и призывает отложить паранойю по этому поводу. «Да, компьютеры лучше справляются с визуальным изображением, чем 99% людей

на планете. Но это не значит, что мы перестанем воображать. Машины передвигаются быстрее людей, но ведь мы из-за этого не перестали ходить. Мы перемещаем огромное количество вещей на огромные расстояния, нам для этого нужны самолеты, лодки или автомобили с более приспособленными для таких задач двигателями, — констатирует Хольц. — Так что мы рассматриваем Midjourney исключительно как двигатель воображения. И лично я считаю нашу нейросеть очень гуманистическим изобретением».

2.2. Как создать изображения с помощью нейросети

Мы заходим на главную страницу сервиса `lexica art` здесь мы можем наблюдать уже примеры.

Созданных при помощи данного сервиса различных изображений и самое интересное если кликнуть на какое-либо изображение мы увидим текстовую подсказку с помощью которого генерировал данные изображения можем и скопировать. Для дальнейшего использования скопировать URL на данный пример.

Данный пример можно открыть в редакторе и использовать его как основу для новой генерации, либо осуществлять поиск по библиотеке сгенерированных изображений. Можно лайкнуть понравившиеся нам пример и он добавится в отдельный раздел на сервисе. Генерируется сразу же 4 варианта изображений вот можем их посмотреть. Давайте его скопируем чтобы потом его можно было использовать закрываем что тут еще можно искать по библиотеке изображений ведя какой-то запрос поле поиска например мы поищем запрос веб-сайт нажимаем поиск и вот на наш такие вот результаты можно открыть его посмотреть.

Также можно осуществлять поиск по библиотеке по уже существующему изображению для этого нужно назначить нажать на значок такой картиночки выбрать необходимый файл на компьютере и вот у нас нашлись такие фото. Для использования сервиса необходимо пройти процедуру регистрация для этого нажимаем `GetinstPetstarted` кнопку. Можно зарегистрироваться через аккаунт Google либо используя адрес электронной почты. Теперь мы можем использовать сервис копирую вставляем нашу ссылочку которую мы скопировали например и давайте попробуем нажимаем `OpeninEditor` можно сгенерировать по примеру изображения. Как видите здесь сразу добавляется тексту подсказка

и вот изображение, на основе которого будем генерировать новые. Нужно нажать на кнопку Generate. Возвращаемся назад вот можно каждое изображение посмотреть более крупном формате. Вот экспорт лист стайл, тогда мы найдем все похожие изображения библиотеки сервиса лексика арт. Понравившись изображению можно вот поставить лайк, тогда она добавится в отдельный раздел Openinitier. Есть дополнительные параметры с правой стороны. Можно указывать размер итогового изображения и есть еще дополнительные опции с этим, нажимаем на кнопку Generate происходит достаточно быстро и у нас создание нового изображения здесь вот сразу же генерирует 4 варианта можно посмотреть изображение, увеличить его. Но новый пример для запроса для генерации либо скачать его себе на компьютер все созданные изображения сохраняются в разделе History вот здесь вот показано 4 изображения бесплатно вы можете создать до 100 изображений то есть на самом деле генерации будет не 100 а в четыре раза меньше поскольку генерируется за раз 4 изображения в разделе Likes.

Like отображаются те варианты примеров которые мы лайкнули.

Генерации здесь вот еще кроме поля текстов запросу есть еще дополнительное поле куда можно добавлять те элементы которые мы хотим исключить с итогового изображения, например какие-нибудь там очки и галстук еще что-то добавляются. Все запросы делаются на английском языке.

Данный сервис использует собственный генератор LexicaAperture v2.

Создать любое изображение в нейросети можно зная простой синтаксис специальных запросов либо хотя бы английский язык. Можно даже не знать ни того, ни другого и пользоваться Google переводчиком.

На практике это выглядит так: вы пишете то, что хотите увидеть и нейросеть генерирует вам картинку или серию изображений, содержащих ваш запрос. Принцип работы базируется на огромной выборке случайных фото, с помощью которой происходит обучение искусственного интеллекта. Размер выборки поистине огромен и исчисляется миллиардами экземпляров,

поэтому система распознает практически любые текстовые запросы и генерирует на их основе уникальные изображения.

Почему практически любые, а не все? Дело в том, что здесь как всегда вмешиваются языковые различия в разных лингвистических группах.

Например, при переводе из одного языка в другой с помощью того же онлайн-переводчика может происходить калькирование из-за чего смысл запроса может измениться, и соответственно выдача готового результата тоже будет некорректна. Как правило, проблемы возникают при использовании фразеологизмов, имен собственных, а также упоминании каких-то мифических существ, присущих только определенной культуре: домовой, вендиго, шайтан и т.д.

Для примера прикрепляем пару изображений, сгенерированных по подобным запросам.

Кот в сапогах



2.3. Создание иллюстрации нейросетью

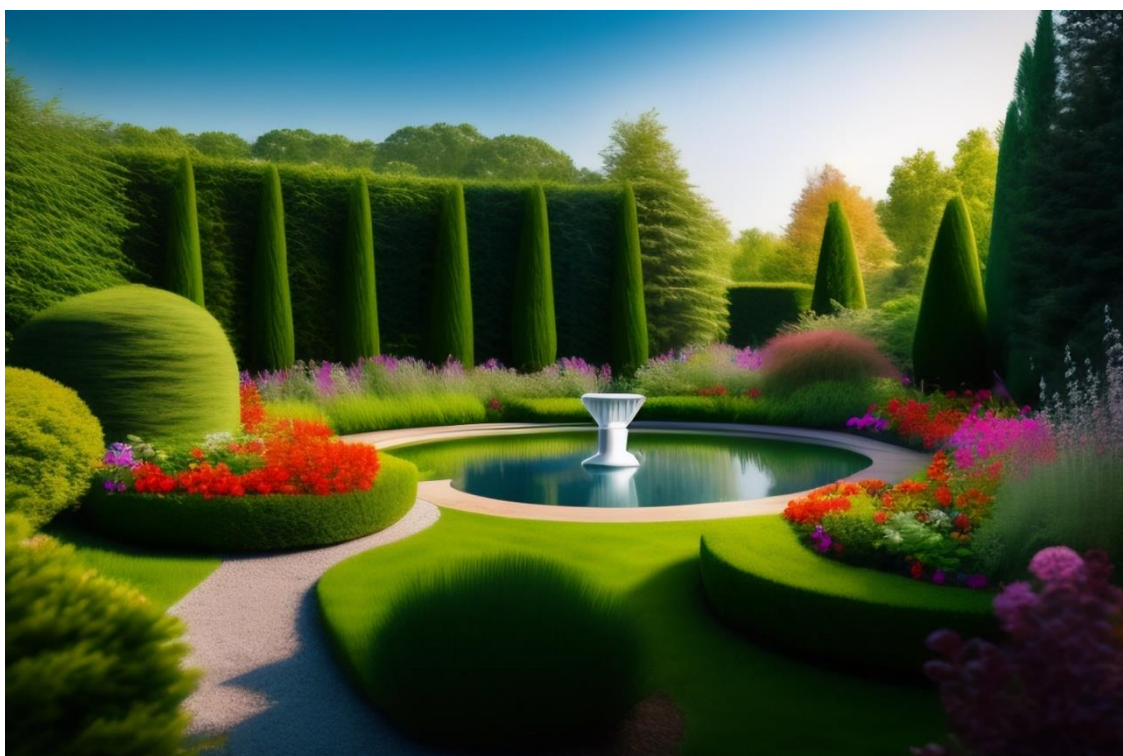
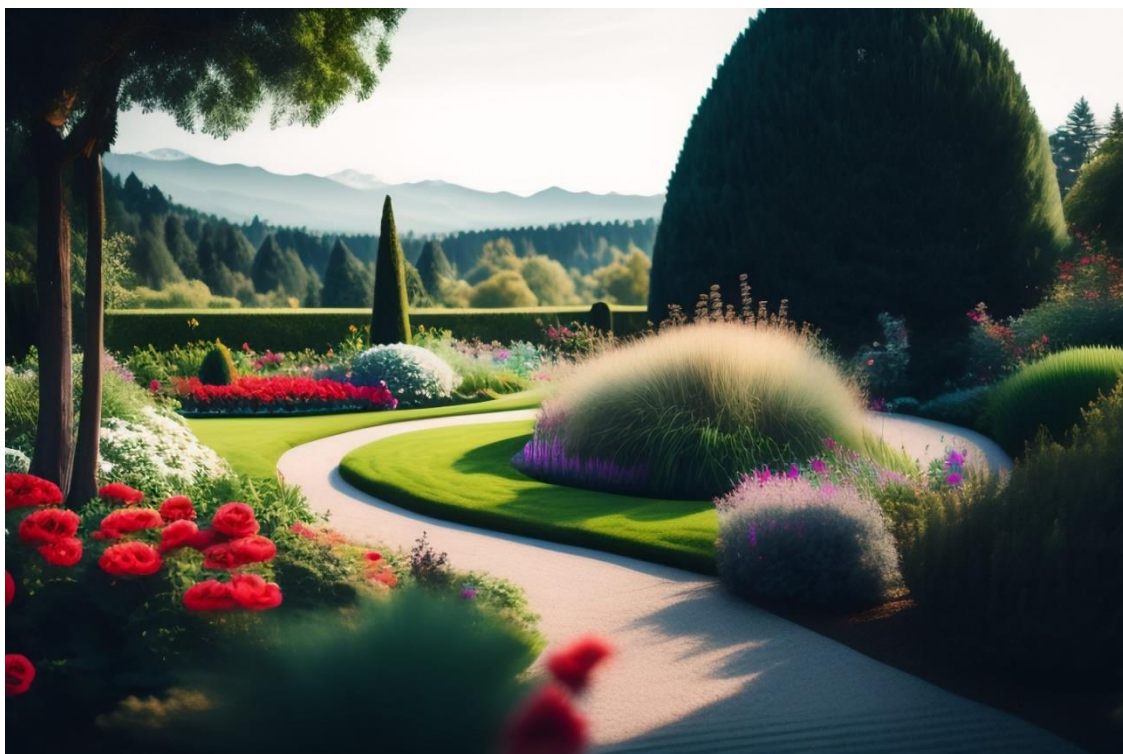
Иллюстрация, сгенерированная нейросетью, во многом не уступает произведению художника или фотографа. Можно создать практически все, что вы можете себе представить и выразить словами: реалистичные фотографии, мультяшные изображения, эскизы и абстрактные пейзажи.

Примеры запросов

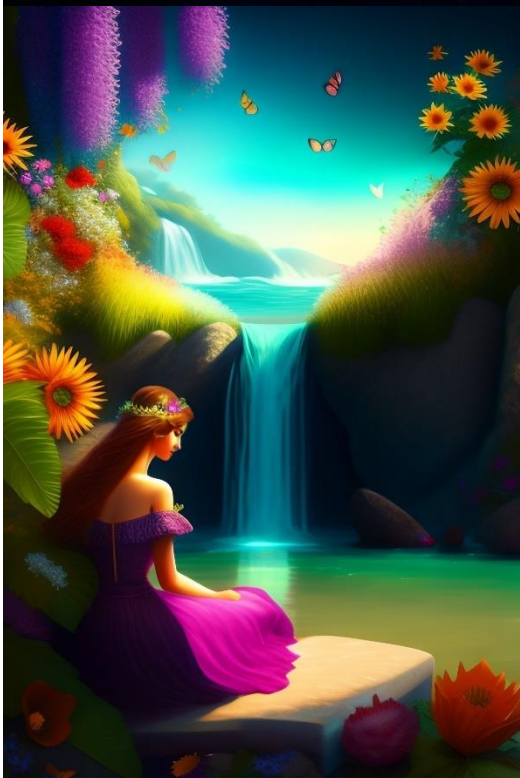
Нужно фото молодой девушки с серебристыми волосами в пушистом сером пальто, которая стоит в заснеженном зимнем лесу



Весенние пейзажи



Летние пейзажи



Осенние пейзажи



Зимние пейзажи



Абстрактные пейзажи

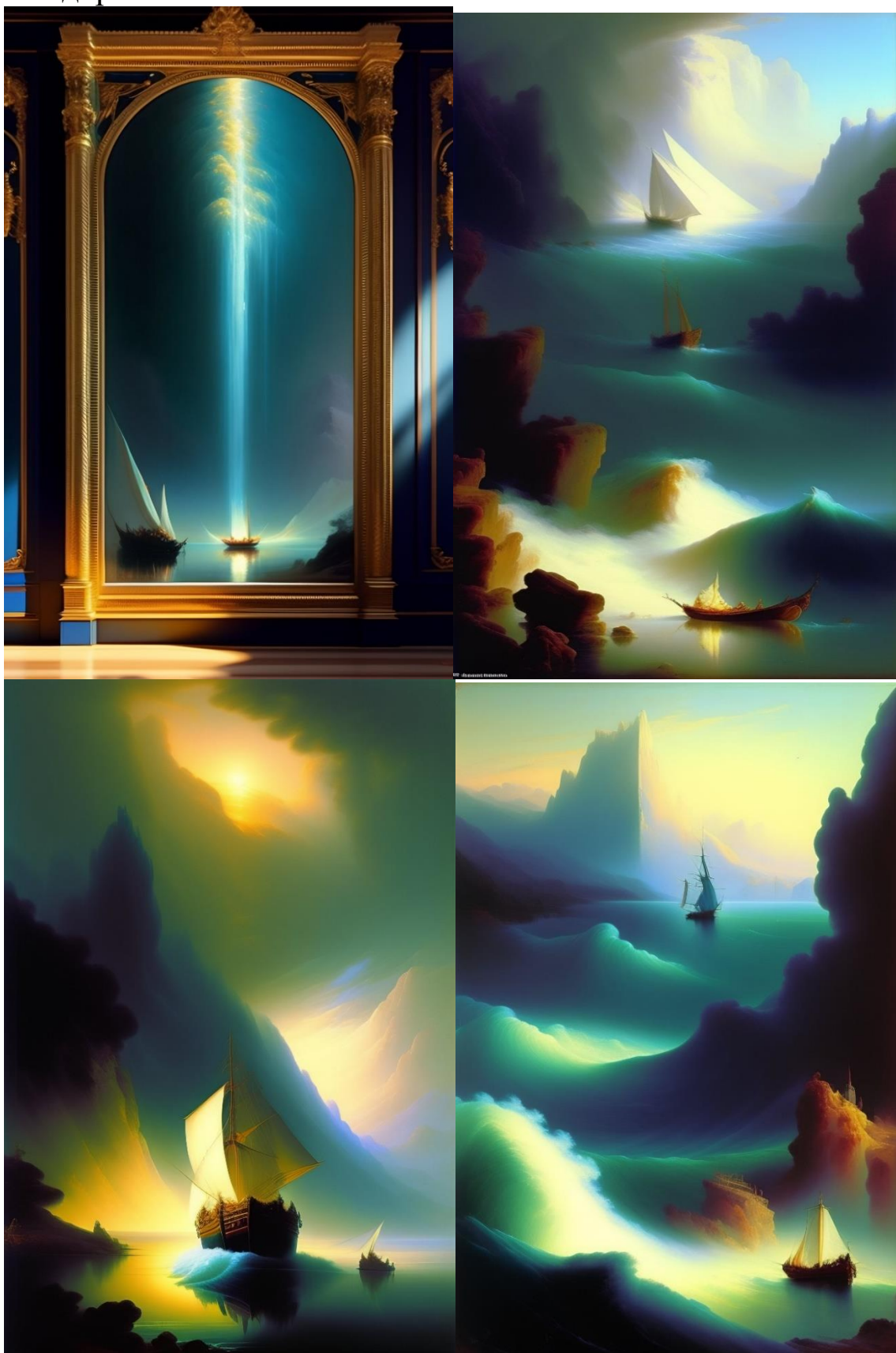


Леонардо да Винчи Мона Лиза





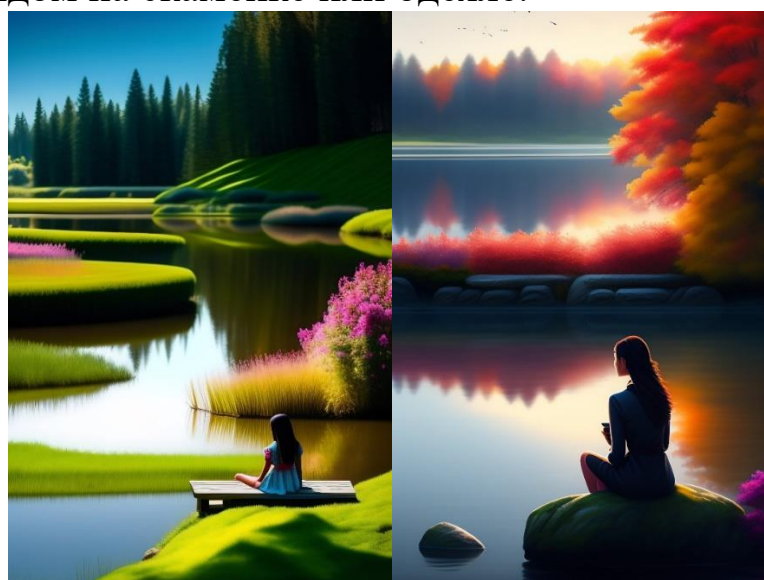
Пандора масло на холсте Айвазовского



Молодая женщина перед полным кадром, смотрящая за зрителем, одетая в короткое прозрачное платье, с ногами в воде ручья, смотрящая на свои ноги, сидящая на берегу, большое дерево позади нее, на заднем плане океан, картина маслом в манере Тициана



У девушки должны быть длинные распущенные волосы и умиротворенное выражение лица. Пруд должен быть окружен сочной зеленой травой и деревьями, а вода должна отражать окружающий пейзаж. Девушка может стоять на берегу пруда или сидеть рядом на скамейке или одеяле.



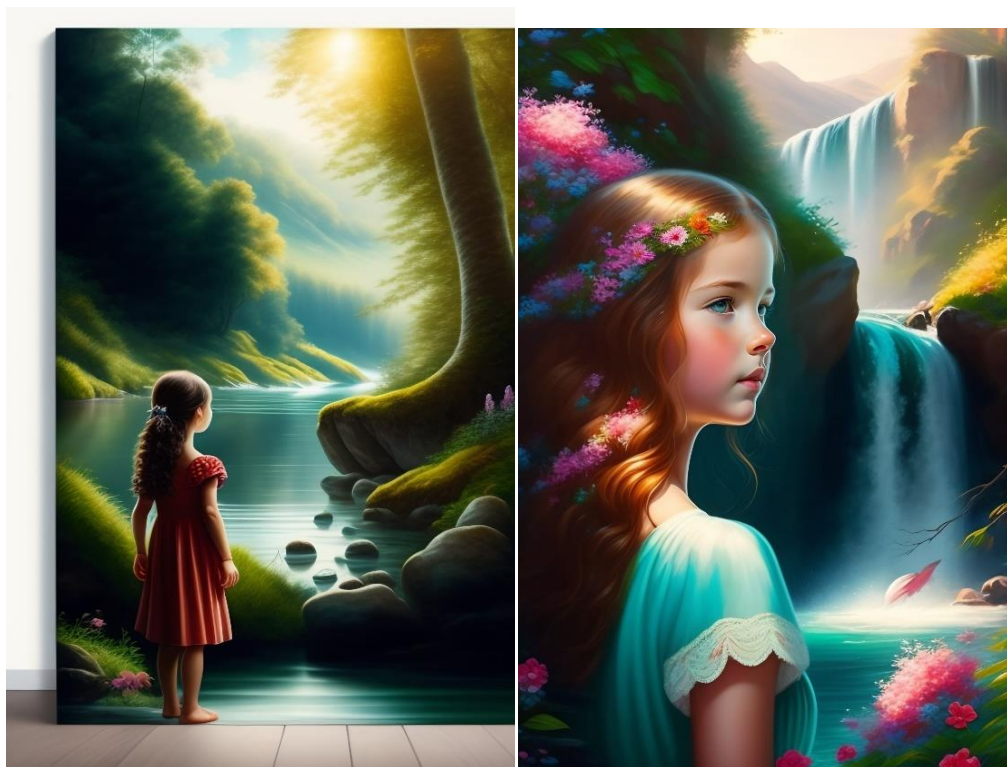
Грустная девушка, лес издалека, деревья, старое белое платье, тени, красная ночь, сидит, тротуар, масло, цветы, звёзды.



Романтическая картина маслом в стиле прерафаэлитов портрет высокой женщины в тонком розовом простом струящемся платье, идущей босиком по крошечной вересковой пустоши с рушащимися замками и зелеными холмами



Молодая девушка с ногами в воде ручья, смотрит на водоток, опирается на берег, за ней большое дерево, на заднем плане океан, матовая живопись в стиле Микеланджело



Женщины в стиле Рафаэло Санцио стоят на улице



Время в стиле Сальвадора Дали



Группы счастливых людей в стиле Сальвадор Дали



Девушка Вермеера с вариацией жемчужной серьги перед окном с видом на золотой век голландских домов

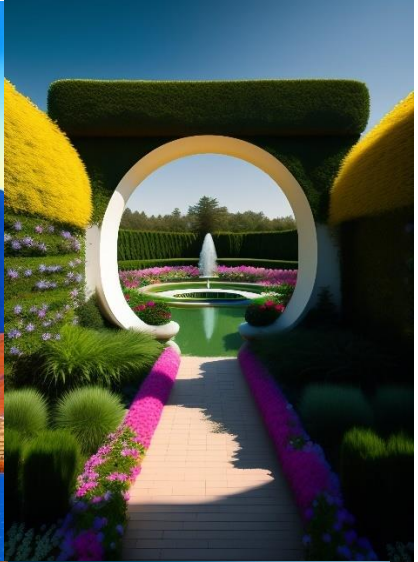
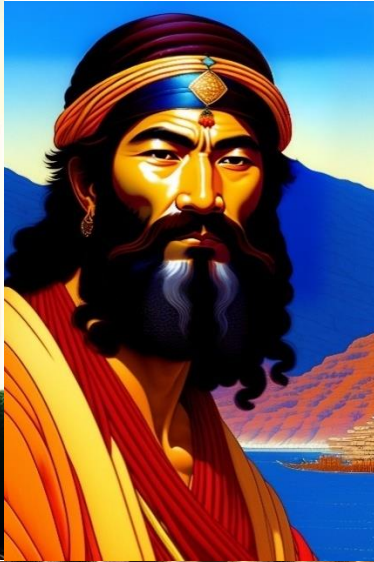
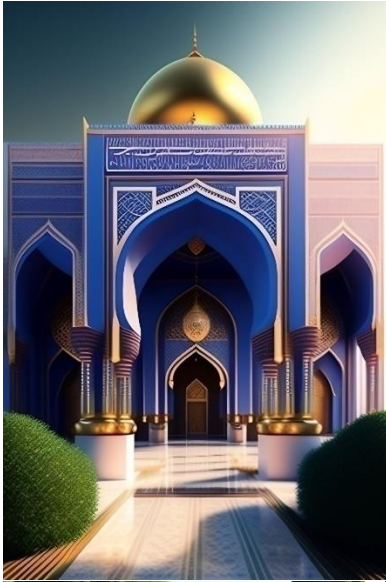


Барышня играет на лютне, светлые волосы, окно справа, гобелен на задней стене, как у Вермеера

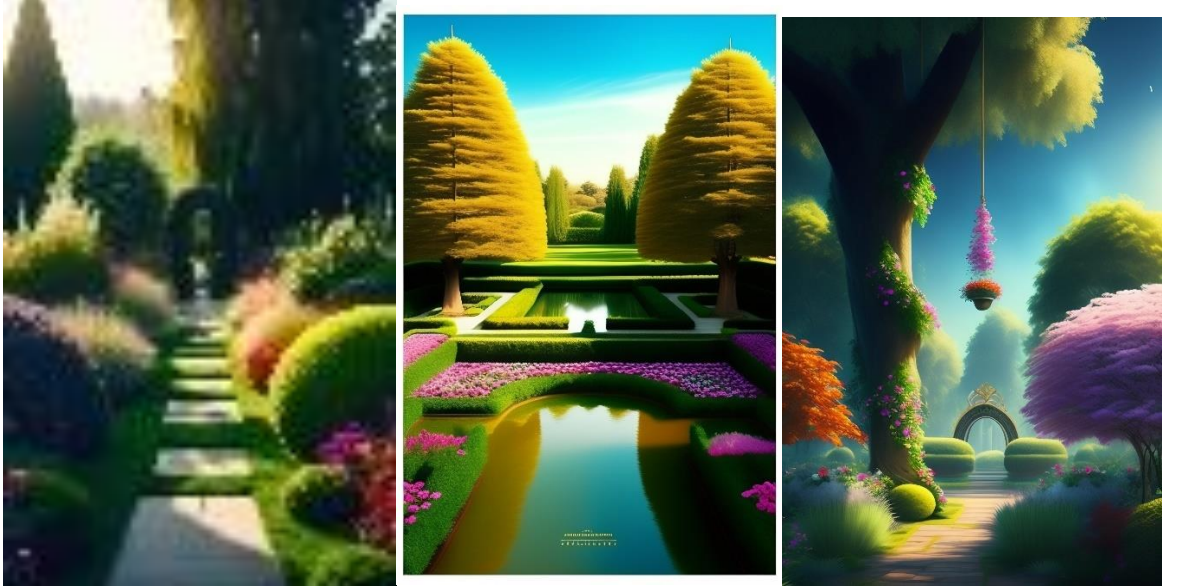
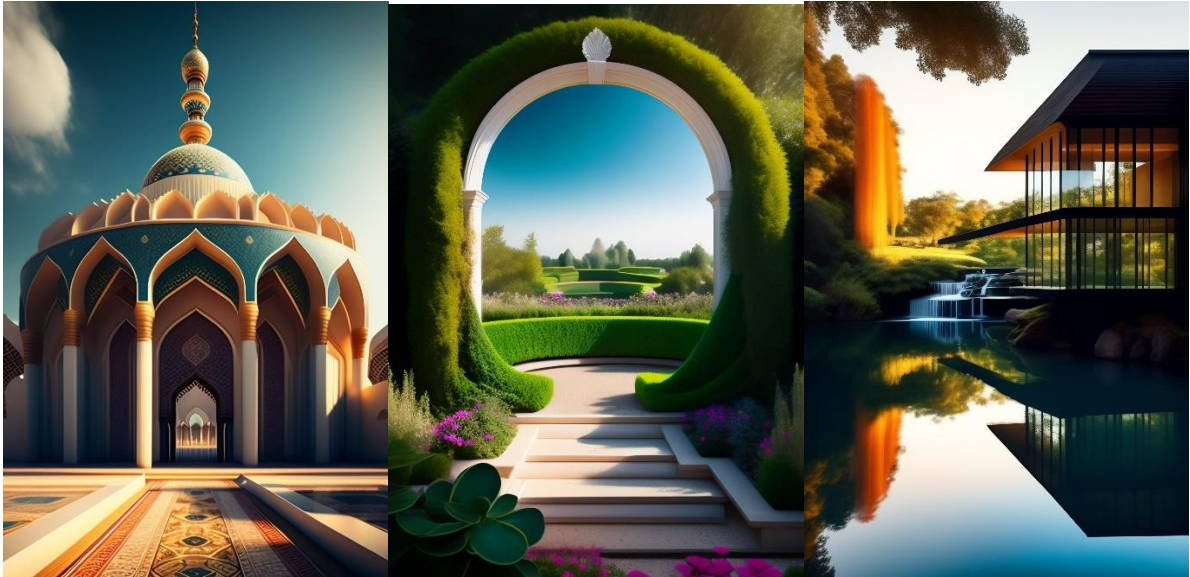


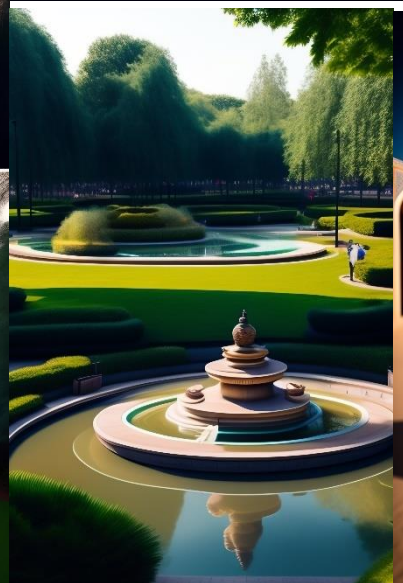
Как видит нейросеть Узбекистан

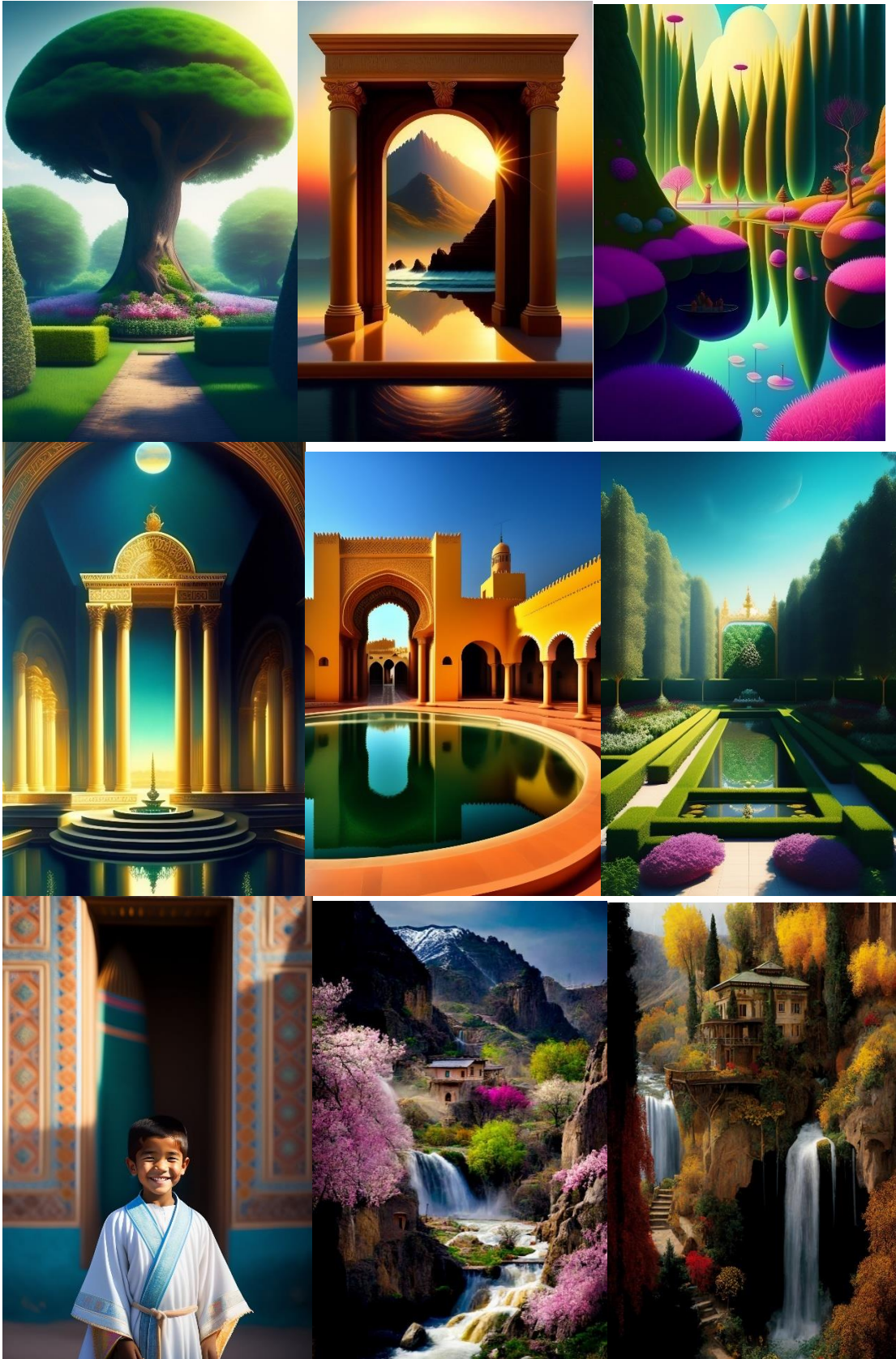




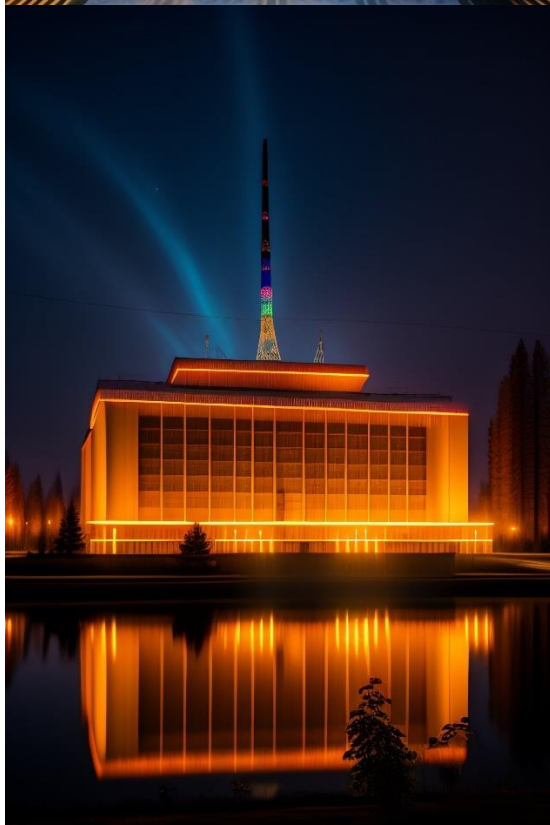
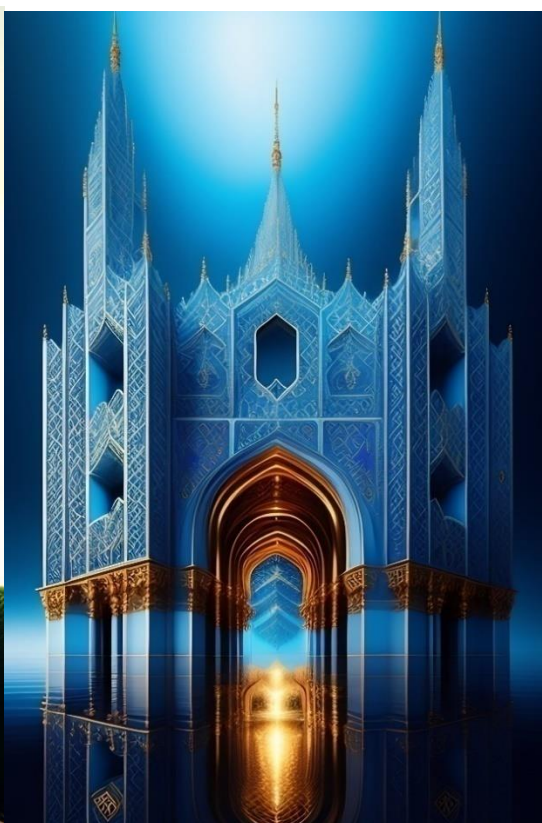








ТАШКЕНТ



Именно так видит наш район и сквер Амира Тимура одна из нейросетей нашего времени(сгенерировано и нарисовано с нуля по запросу)





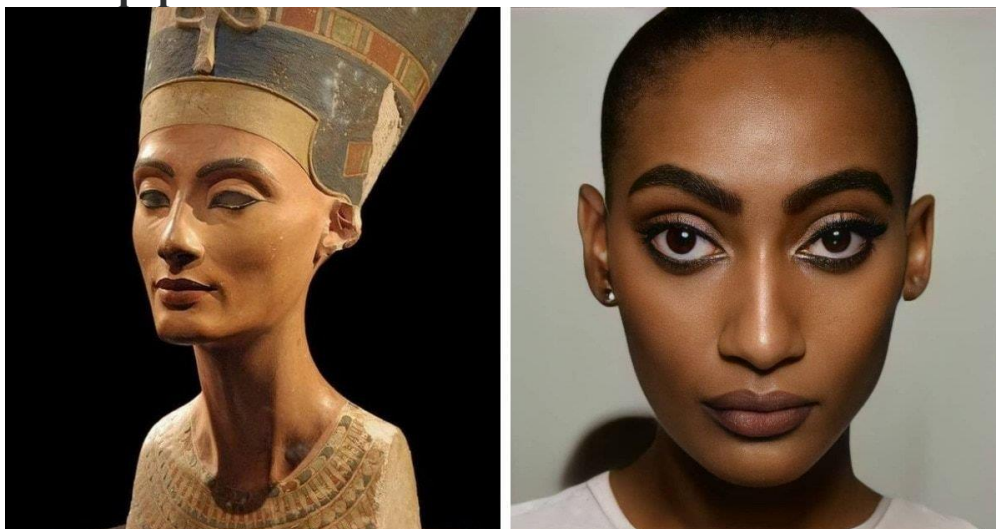
Природа Узбекистана



Как выглядели бы исторические личности, будь они нашими современниками

Всегда интересно заглянуть в прошлое, и узнать, как жили люди в далекие времена, как они выглядели, во что одевались. К сожалению, дойдя до нашего времени сохранилось не много реликвий, на которых изображены исторические персоны, кто-то из них запечатлен на гипсовых бюстах, кто-то на картинах. Но глядя на исторические полотна прошлого, слабо представляется как же они выглядели на самом деле. Но к счастью, в наше время технологий, оживить некогда живших людей, совсем не проблема.

1. Нефертити



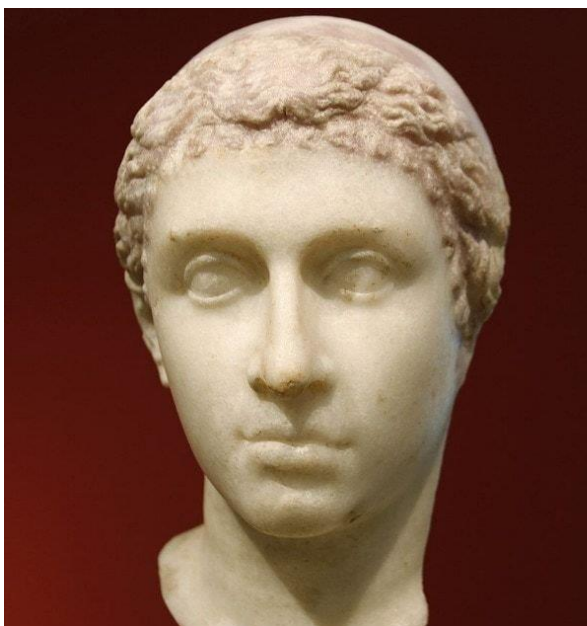
Приблизительная дата рождения: 1370 год до нашей эры.

2. Иоганн Себастьян Бах



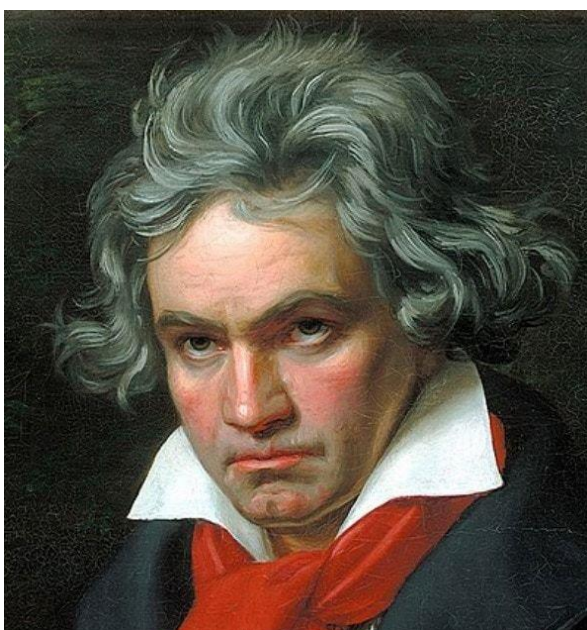
Год рождения: 1685.

3. Клеопатра



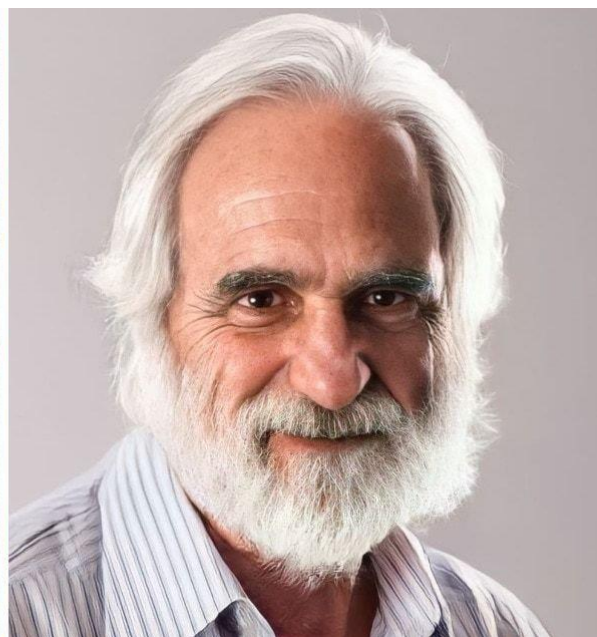
Приблизительная дата рождения: 69 год до нашей эры.

4. Людвиг ван Бетховен

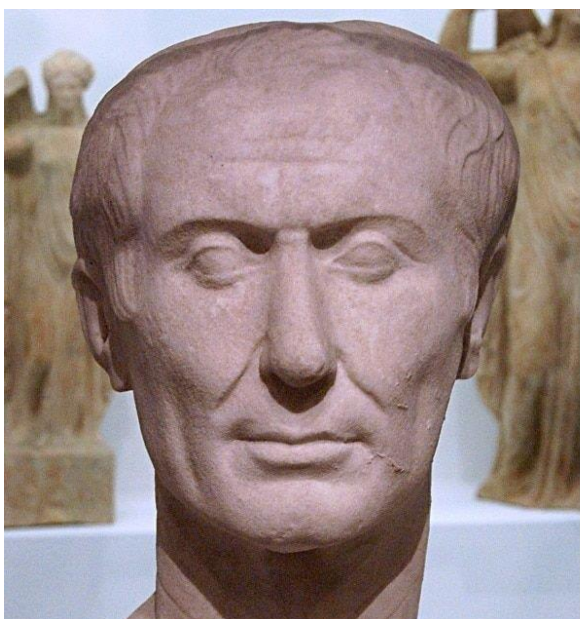


Год рождения: 1770.

5. Леонардо Да Винчи



6. Гай Юлий Цезарь



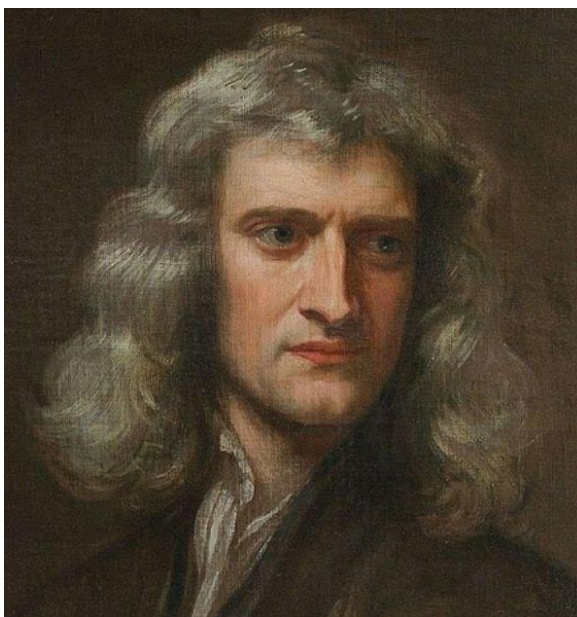
Дата рождения: 100 год до нашей эры.

7. Жанна д'Арк



Приблизительная дата рождения: 1412 год.

8. Исаак Ньютон

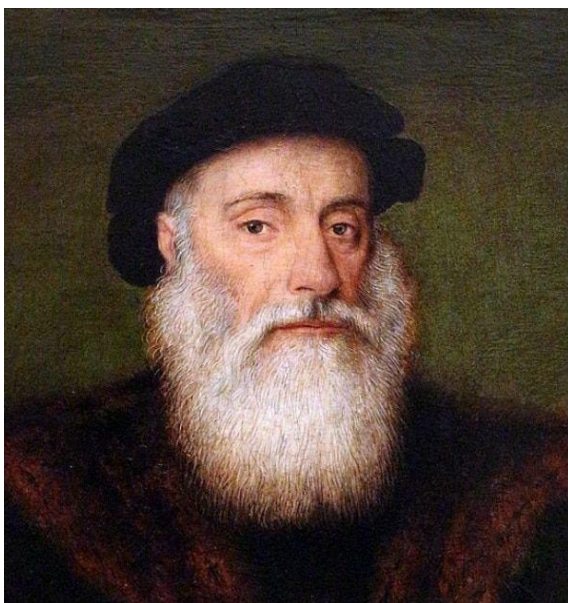


9. Антонио Вивальди



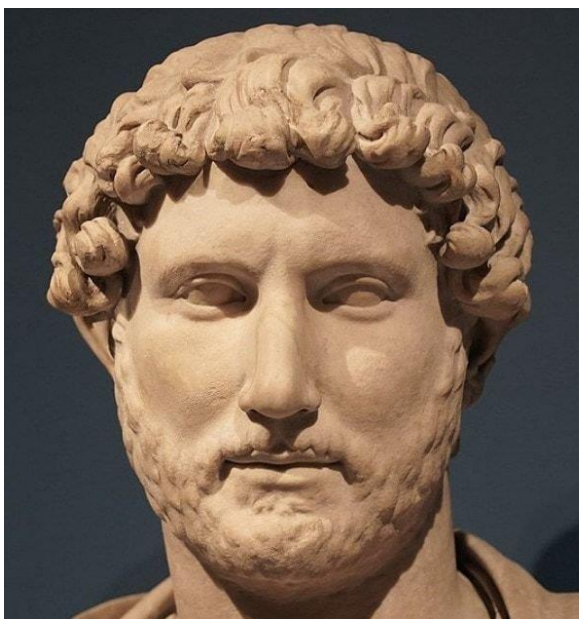
Год рождения: 1678.

10. Васко да Гама



Год рождения: 1469.

11. Адриан



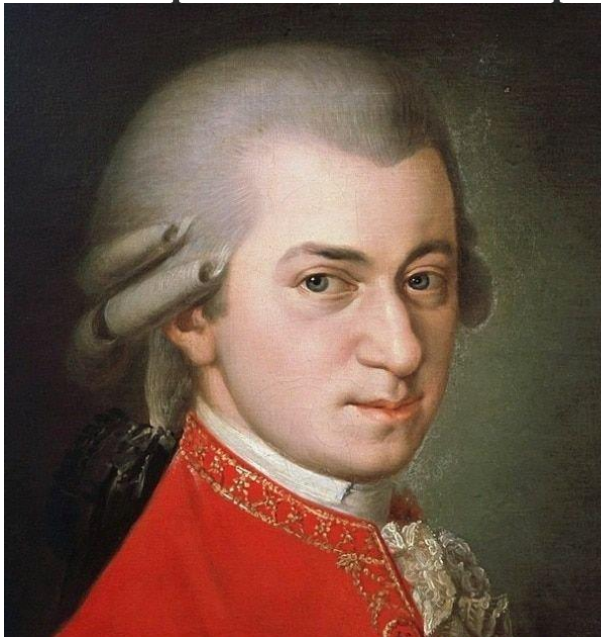
Дата рождения: 76 год нашей эры.

12. Микеланджело



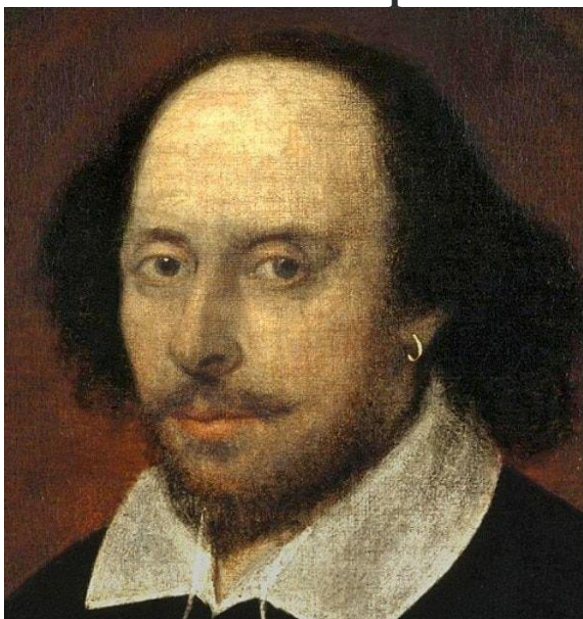
Год рождения: 1475.

13. Вольфганг Амадей Моцарт



Год рождения: 1756.

14. Уильям Шекспир



Приблизительная дата рождения: 1564 год.

Искусственный интеллект сгенерировал страны в обличье детей





Нейросеть сгенерировала города Узбекистана в человеческом обличи

Midjourney показала образы людей, отражающие исторические корни жителей Термеза, Бухары, Ургенча, Нукуса и других городов Узбекистана.

Нейросеть отразила образ девушек Узбекистана в Термезе, Бухаре, Хиве, Ташкенте и Фергане с колоритными орнаментами на платке и одежде, крупными украшениями, а на фоне изобразила символические объекты с историческими элементами.

При этом Ургенч, Нукус и Самарканд показан в образах стариков, отражающих исторические корни жителей этих регионов.

источник <https://pr.uz/kak-nejroset-uidela-uzbekistan/>





ТОШКЕНТ

ТАШКЕНТ



ФАРҒОНА

ФЕРҒАНА



ТЕРМИЗ

ТЕРМЕЗ



БУХОРО

БУХАРА



УРҒАНЧ

УРҒЕНЧ



НУКУС

НУКУС



ХИВА

ХИВА



САМАРҚАНД

САМАРҚАНД

Ну а тут есть забавный пару примеров того, как нейросеть иллюстрирует некоторые узбекские народные присказки и поговорки. Получилось весьма интересно.

Аркан сразу заметят: души врага ватой.



Богатство детей — отец и мать.



Была бы голова цела, тюбетейка найдётся



Лучше умереть стоя, чем жить на коленях



Не хлопай дверь, в которую вошёл.



Глава3. ПРИМЕНЕНИЕ СОВРЕМЕННЫХ НЕЙРОСЕТЕЙ ПРИ ГЕНЕРИРОВАНИЯ ТЕКСТА

Рассмотрим, как нейросети генерируют текст, какие есть сервисы на рынке, какой контент они напишут лучше, а какой хуже.

Сегодня нейросети способны генерировать осмысленный текст за несколько секунд. Технология представлена в ряде сервисов, которые позволяют заметно ускорить производство текстового контента определённого формата.

3.1.Как нейросеть генерирует текст

Нейросеть — это алгоритм искусственного интеллекта, если точнее — метод машинного обучения. Алгоритм, используя данные, должен решить определённую задачу, в процессе работы он учится находить закономерности в этих данных и автоматически их обрабатывать. Алгоритмы представляют данные в виде чисел и взаимодействуют с ними математически.

Востребованная сегодня задача — это обработка естественного языка, или NLP (NaturalLanguageProcessing). Простыми словами, это технология, которая позволяет машинам “понимать” человеческие языки. В задачу NLP входит работа с текстом. Нейросети способны устанавливать смысловые связи между словами с учётом их контекста, а также генерировать новый осмысленный текст.

На основе нейросетей работают сервисы, которые создают различный текстовый контент. Они напишут электронную рассылку, отзыв о товаре, введение и отдельные абзацы для статьи. Могут сгенерировать идеи и релевантные заголовки и даже писать текст по маркетинговой модели AIDA и PAS.

Прежде чем перейти к конкретным примерам, кратко расскажу, как нейросети генерируют текст и извлекают из него смысл. Лучше всего с данной задачей справляются нейросети-трансформеры. При обработке текста трансформер представляет каждый его элемент (токен) в виде вектора, группы чисел, и работает с ним отдельно, производя математические операции. Это позволяет нейросети учесть порядок слов и их взаимное влияние друг на друга и таким образом, определить смысловое содержание текста.

На основе трансформеров создаются гигантские языковые модели, перед ними ставят задачу языкового моделирования: нейросети дают текст, а она должна его продолжить. Чтобы научиться этому, нейросеть запоминает, как устроен язык, и узнаёт разные факты о внешнем мире, для этого используют огромные массивы текстов из интернета, которые охватывают весь спектр тем. В результате такая модель способна решить новую текстовую задачу с ходу без дополнительного обучения — это так называемый Few-shotlearning.

Сервисы для генерации текста

Покажем на примере модели ruGPT-3 (GenerativePre-trainedTransformer 3), как трансформер продлевает предложенный ему текст.

Эти тесты [опубликованы](#) Татьяной Шавриной, одним из разработчиков данной модели.

Получив заправку, где есть любимые фильмы человека, модель сразу сгенерировала несколько вариантов фильмов похожих жанров.

Заправка: Мои любимые фильмы — “Матрица”, “Олдбой”, “

“ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Крёстный отец”, “Игра в имитацию”, “ОстинПауэрс”.

ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Спасение рядового Райана”, “Нечто”, “Затура”.

ruGPT-3 Large: Мои любимые фильмы — “Матрица”, “Олдбой”, “Эквилибриум” и др.

Текст для заправки кавычкой не закрыли, поэтому модель генерировала имена собственные.

Пример более сложной задачи — генерация отзыва, где надо вести себя, как человек.

Заправка: “Напиши отзыв о стиральной машине. Данная модель стиральной машины”

ruGPT-3 Large: “Напиши отзыв о стиральной машине. Данная модель стиральной машины оснащена встроенной функцией самоочистки. В конструктивном решении стиральной машины предусмотрено устройство автоматического затирания воды, которая способна удалять накипь и прочие загрязнения даже

в труднодоступных местах. Стоит отметить, что после процедуры чистки, во время отжима появляется специфический запах. У некоторых моделей, в результате эксплуатации, он сохраняется в течение всего срока службы стиральной машины.”

Затравка: Напиши отзыв о фильме. Фильм "1+1"

ruGPT-3 Large: Напиши отзыв о фильме. Фильм "1+1" — очень нравится, хотя и смотреть его стоит не каждому. Актёрский состав в нём великолепен. Смотрится без особого напряжения и с интересом. В фильме очень много смешных моментов. Так же советую посмотреть фильм "Зелёная книга", также фильм с Дженнифер Лопес и Крисом де Бургом.

В целом модель способна продолжать текст на любую тему и её можно использовать для копирайтинга. Помимо генерации, модель умеет работать с разными текстовыми задачами. Она применима:

- В рекомендательных системах: анализ предпочтений пользователей и предложения релевантного контента.
- Для резюмирования текстов: краткий пересказ с сохранением смысла.
- В анализе настроений текста: классификация на положительные и отрицательные.
- В диалоговых системах (с дообучением на конкретных сценариях, подойдет для бизнес-приложений).

RuGPT-3 обучали на 600 Гб информации, куда вошла коллекция русской литературы, “Википедия”, новостные ресурсы и сайты с вопросами и ответами. Самая крупная версия модели имеет 13 млрд параметров: чем больше их число, тем более сложные закономерности модель сможет выучить.

Для тестовой генерации есть **демоверсия**. Чтобы использовать модель полноценно, придётся зарегистрироваться как юрлицо на платформе **ML space**. За плату модель получится гибко настроить под необходимую задачу. Если есть навыки в разработке это можно сделать самому, другой вариант обратиться за помощью к специалистам из сбера.

Русскоязычная модель является аналогом трансформера **GPT-3** от разработчиков OpenAI. Она усвоила 600 Гб текста и имеет 175 млрд параметров. Модель можно использовать практически в любых задачах, связанных с текстом на английском языке. GPT-3

— одна из самых больших нейросетей для генерации текста. Статистика [показывает](#), что её используют более 300 приложений от коммерческих до игровых, а в день она в среднем генерирует 4,5 миллиарда слов.

Разработчики предлагают несколько вариаций моделей, подходящих под разные задачи, есть возможность настроить собственную. Нейросеть генерирует текст на английском, платить придётся за каждые 1000 токенов это приблизительно 750 слов, цена зависит от модели, начинается от \$0,0008. Однако из некоторых стран регистрация не пройдёт, придётся использовать другую почту и зарубежный номер телефона.

Гораздо проще использовать сервисы-генераторы, они сразу готовы к работе и рассчитаны на обычного пользователя. Расскажу о тех, что могут писать текст на русском.

На основе GPT-3 действует [Jasper AI](#) — это сервис по созданию текстового контента для сайта, личного блога, социальных сетей и много другого. Jasper поддерживает 25 языков, в том числе и русский, его тексты оптимизированы под SEO и не содержат плагиата. Сервис имеет свыше 50-ти шаблонов, каждый нацелен на конкретное применение.

Например, есть шаблоны, которые пишут текст по маркетинговым моделям AIDA и PAS.

На скриншоте работа с шаблоном AIDA. Слева вносится имя продукта и его краткое описание и выбирается характер текста. В данном случае выбран «остроумный». Справа — результат генерации. Процесс занимает несколько секунд.

По такому принципу работают все другие сервисы-генераторы: вы даёте краткую информацию того, что хотите получить, и нейросеть создаёт текст, отталкиваясь от этих данных.

Другие навыки Jasper подойдут под написание поста, письма, генерации идей, заголовков для статьи и т.д. Минимальный тарифный план стоит \$29 в месяц и имеет ограничение в 20 тысяч слов. Можно получить 5 дней бесплатного использования, но потребуется привязка карты.

Большой набор навыков для написания разного контента предлагают платформы [Outranking](#), [Writesonic](#) и [Rytr](#). Все зарубежные, но поддерживают русский язык.

Outranking выделяется тем, что имеет гораздо больше инструментов для SEO, чем его аналоги. Распространяется платно, месячная подписка от \$49 даёт сгенерировать 60 тыс. слов и предлагает 10 текстовых материалов для SEO-настройки. Годовая от \$499, в месяц выходит \$41.58 с тем же лимитом возможностей.

Создатели Writesonic отмечают, что модель была обучена на качественных текстах ведущих брендов. Сервис дают испытать бесплатно, предоставляя приблизительно 5 тыс. слов, подписка от \$15 в месяц с ограничением в 50 тыс. слов.

Rytr, похоже, самый экономичный вариант, бесплатный тарифный план включает 5 тыс. символов в месяц, а подписка с \$9 в мес. позволяет генерировать 50 тыс. знаков.

[Gerwin AI](#) по функциональности может сравниться с Jasper. При этом у Gerwin есть шаблоны для описания товаров на сайты российских маркет-плейсов. Подписка за месяц идёт от 1900 рублей на 100 тыс. символов. Можно приобрести отдельный пакет, минимальный стоит 330 рублей и включает 10 тыс. символов.

Ещё одна платформа — [AIWriter](#). На выбор даётся 15 категорий для создания разного контента. При регистрации по почте предлагается 7 дней пробного доступа. Месячная подписка требует 750р., годовая 4500р., ограничений по количеству слов нет.

[ReText.AI](#) — инструмент для рерайта русскоязычных текстов. Нейросеть позволяет перефразировать исходный текст и получить новый и уникальный, полностью сохранив смысл. Есть бесплатная, но ограниченная версия, весь инструментарий продаётся по ежемесячной подписке от 400 рублей с ограничением в 100 тыс. символов.

Сервис [Copy Monkey.ai](#) предлагает для генерация более 10 видов контента. Это статьи, лендинги и разные вариации коротких текстов. Например, карточки товаров для Wildberries и Ozon, отзывы на товары, письма для рассылки.

Для ознакомления можно сгенерировать 10 текстов, они обновляются каждый день. Чтобы получить полную версию сервиса нужно заполнить форму на сайте. Платформа интегрируется по API, вам могут выслать доступ, если оставите заявку через сайт.

Точных расценок и тарифов на странице сервиса пока нет.

Анализ сервисов

Важно понимать, что Jasper и аналогичные ему продукты не панацея, и писать абсолютно весь контент не способны.

Зарубежный портал DDIY, занимающийся аналитикой инструментов для бизнеса, раскрыл преимущества и недостатки Jasper. Согласно их оценке, сервис лучше всего подходит для коротких текстов: постов, писем, заголовков, описания видео, продуктов и т. д. Он подскажет новые идеи, создаст скелет контента, который уже можно наполнить чем-то уникальным.

В статьях от 1500 слов и более текст выходит шаблонным, информация собирается со всего интернета и не проверяется на достоверность, материал придётся вычитывать, проверять на факты и редактировать. По сути, вы потратите столько же времени, если будете писать статью самостоятельно.

Gerwin хорошо сгенерировал короткие тексты. Я задавал основные характеристики продукта или услуги, а нейросеть выдала лаконичное описание, где подчеркнула все преимущества. Текст соответствовал формату, который я выбрал, письму, посту, описанию продукта и т.д.

Со статьей сервис не справился, но немного помог её написать. Для этого я отдельно сгенерировал основные абзацы статьи. В них можно увидеть семантическое ядро темы и понять какие вопросы следует раскрыть. Это позволит быстрее вникнуть в тему и потратить меньше времени на поиски информации. Но сразу брать сгенерированный материал в статью не стоит, его требуется проверить, насколько он достоверен и нет ли тут противоречий.

Например, на темы технологий в генерации было много непонятных терминов. В них бы пришлось разобраться и описать простыми словами, прежде чем добавлять в статью.

Ещё лучше формулировать узкий и точный запрос, иначе нейросеть начнёт расплываться по темам или перескочит на другую, не связанную с запросом. Тогда вы потратите символы, за которые платите в подписке, но не получите нужный текст.

Аналоги Jasper и Gerwin генерируют текст по такому же принципу, поэтому считаю, что эту оценку можно применить и к ним.

Опытом работы с сервисами для генерации поделился Рамиль Каримов, директор по Digital коммуникациям «КРОС»:

«Полноценную статью нейросети написать всё ещё не могут, выходит бессистемный поток слов, но отдельные абзацы получаются довольно сносно. Например, мы использовали «Балабоба» для проведения весёлых конкурсов в социальных сетях – большое поле для креатива и роста Engagementrate. Была надежда, что такие умные сервисы смогут генерировать SEO-тексты и помогать в работе по SERM, но этого не произошло. Много сервисов работают с английским языком, но русский язык сложнее. Его структура, синтаксис и грамматические особенности усложняют процесс генерации».

Сервис CopyMonkey.ai успешно генерирует тексты для интернет-магазинов. Пользователи отмечают, что нейросеть может создавать сотни и тысячи уникальных описаний товаров в месяц. Сервис создаёт текст в разы быстрее и дешевле чем копирайтер.

Выводы: кратко о сервисах для генерации текста

Экспертную статью с помощью сервисов для генерации написать не удастся. Можно генерировать отдельные абзацы, узко формулируя тему. Они помогут понять, какие вопросы нужно осветить в вашей теме, чтобы её раскрыть. Это экономит время, поскольку вы знаете, что надо искать.

Лучше всего сервисы справляются с короткими текстами. Это посты в соцсети, описания товаров, отзывы, заголовки, электронные письма и т.д. Такой материал получится создавать быстро и в большом объёме.

На рынке есть ряд сервисов, позволяющих создавать коммерческие тексты.

Модель GPT-3 генерирует текст на английском и подходит для разных текстовых задач. Однако из России регистрация не проходит, придётся использовать другую почту и зарубежный номер телефона.

ruGPT-3 работает с русским языком и может продлить текст на любую тему и решить другие задачи, связанные с текстом. Чтобы её использовать придётся зарегистрироваться как юрлицо на платформе ML space. За плату модель получится гибко настроить под конкретные задачи.

Среди зарубежных сервисов, генерирующих текст на русском языке, самый функциональный — Jasper, он имеет свыше 50-ти шаблонов для текстов разного типа.

Схожий список текстового контента для генерации предлагают сервисы Outranking, Writesonic и Rytr. Outranking имеет гораздо больше инструментов для SEO, чем его аналоги.

Далее список платформ, которые работают с русским языком и принимают к оплате российские карты.

Gerwin AI имеет более 70-ти навыков копирайтинга и по функциональности может сравниться с Jasper.

AIWriter на выбор даёт 15 инструментов для создания текстов под разные задачи.

SoryMonkey.ai генерирует более 10 видов контента. Это статьи, лендинги и вариации коротких текстов. Изначально сервис специализировался на создании описаний товаров для маркетплейсов и собрал положительные отзывы.

ReTexAI позволять только рерайтить русскоязычные тексты.

Как написать интересный текст с помощью нейросетей

В нашем предыдущем материале мы познакомили вас с онлайн-сервисами, построенными на базе нейросетей, которые позволяют обрабатывать изображения, находить на них закономерности и распознавать образы. Преобладающее большинство таких сервисов используются для развлечения и это нормально — эксперты уверены, что нейросети все еще находятся на начальном этапе развития, нуждаются в популяризации и массово будут использоваться лишь в будущем. Но есть сфера, где искусственный интеллект вполне себе может помочь даже профессионалам ужесегодня. Это работа с текстом и мы собрали для вас подборку таких сервисов.

Самое сложное при составлении списка подобных сайтов — это вычленив из них те, что построены именно на нейросетях. Все дело в том, что текст, в отличие от изображения, представляет собой намного более простой для обработки массив данных. Поэтому компьютеры научились работать с ним еще до активного распространения алгоритмов искусственного интеллекта, которым требуются достаточно серьезные вычислительные мощности.

Перевод с иностранных языков

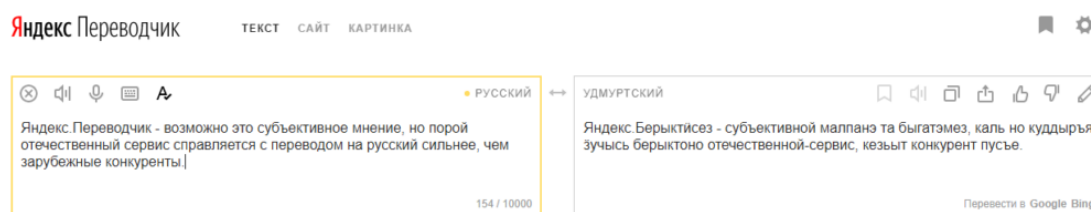
Представлять особо онлайн-переводчики смысла, возможно, нет. Каждый, кто сталкивался с необходимостью перевода с иностранного языка наверняка скачивал на свой гаджет какое-либо из соответствующих приложений, либо открывал сайт одного из самых популярных переводчиков. Единственный момент, который вы могли пропустить, это изменение алгоритма их работы — переход на обработку информации с помощью нейросетей. Например, GoogleTranslate работает так с осени 2016 года, а Яндекс.Переводчик — с сентября 2017-го.

На всякий случай, мы оставим вам ссылки на некоторые из таких сервисов:

GoogleTranslate — один из мировых лидеров. Предлагает огромное количество языков для перевода. При установке на телефон позволяет сохранять языковые пакеты и пользоваться сервисом офлайн. Распознает надписи на иностранных языках и автоматически их переводит прямо на картинке.

Яндекс.Переводчик — возможно это субъективное мнение, но порой отечественный сервис справляется с переводом на русский сильнее, чем зарубежные конкуренты.

Translate.Ru — когда-то офлайн-словари PROMT были безусловным лидером перевода в нашей стране. Они устанавливались на персональные компьютеры с дисков и позволяли прикоснуться к чуду. Сегодня сервис живет в онлайн и ищет свою конкурентную нишу.

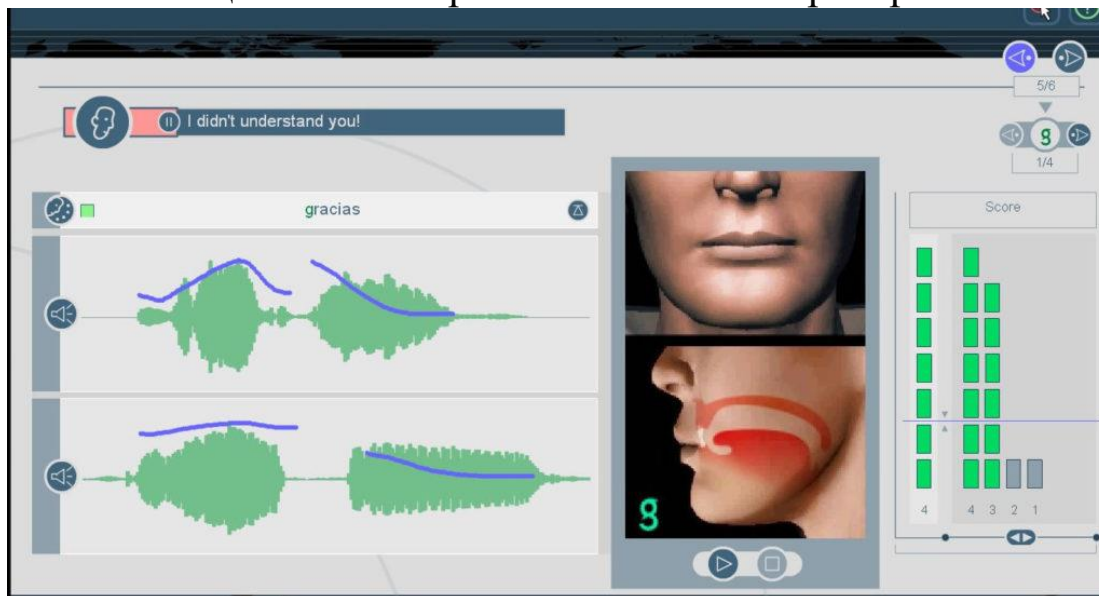


Преимущество переводчика от Яндекса — возможность работать с языками малых народов России.

Распознавание звука и перенос в текст

В сравнении с английским языком разработчики «распознавателей» русского языка сталкиваются с дополнительной проблемой. В русском языке большую роль играют окончания слов, которые часто «съедаются» при разговорной речи. Если люди понимают значения предложений, используя контекст, то машинные алгоритмы сталкиваются с проблемой и часто выдают

абракадабру. И, тем не менее, распознавание речи развивается потрясающими темпами. Если еще несколько лет назад узнавание человеческой речи не превышало нескольких процентов, то сегодня компьютер спокойно, даже в достаточно шумной атмосфере правильно распознает большую часть «скормленных» ему слов. И вот с помощью каких сервисов это можно проверить.



Распознавание звуков как таковых — задача давно решенная. Для машины намного сложнее улавливать контекст речи и делать благодаря этому меньше ошибок

Speechrad — онлайн-блокнот основан на алгоритмах Google. Бесплатный.

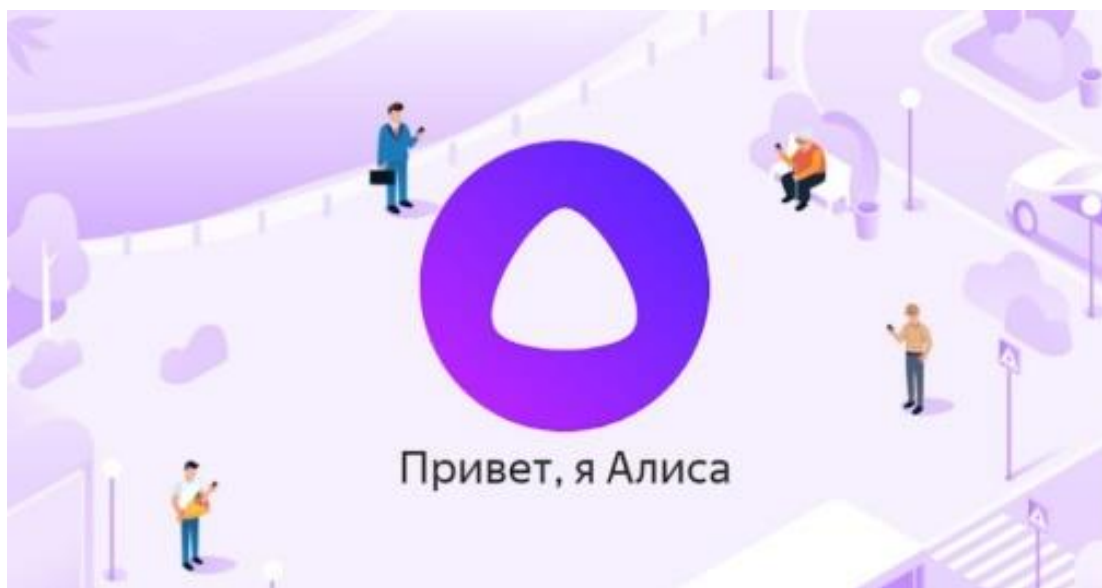
Speechlogger — возможна установка приложением для браузеров.

Speechnotes — просто надиктовываете слова и расставляете знаки препинания. Голосом, на клавиатуре или мышью, выбирая справа на экране.

Dictation.io — еще один блокнот. Позволяет красиво оформить набранный голосом текст и опубликовать.

Если вы разработчик программного обеспечения, то наверняка знакомы с подключаемыми API-сервисами от ведущих поставщиков технологий распознавания голоса. Именно благодаря им работает большинство онлайн-блокнотов и приложений в наших телефонах. Например, Яндекс Cloud — сервис от российского поисковика. При подключении в сторонних приложениях он платный. Стоимость лицензии зависит от количества обращений и в среднем составляет 200 рублей за 1000 запросов или 60 копеек за

1 минуту. Попробовать его бесплатно можно в любом из сервисов Яндекса. Например, в той же «Алисе».



Новые друзья человека

Сделают ли виртуальные ассистенты жизнь проще

Анализ и изменение текстов

Существует множество сервисов, рассчитанных на работу с текстовыми массивами для улучшения поисковой выдачи, упрощения или, наоборот, создания более сложных текстов. Приведем несколько примеров.

Морфер — веб-сервис предназначен для склонения по падежам слов и словосочетаний на русском и украинском языках. Кроме того, он умеет определять пол по сочетанию фамилии, имени и отчества, согласовывать единицы измерения с числом, расставлять ударения, формировать «суммы прописью» и многое другое. На первый взгляд некоторые функции кажутся избыточными, но они крайне полезны, когда вам необходимо обработать большие массивы данных, например, таблицу с несколькими тысячами строк, в каждой из которых необходимо проставить «сумму прописью».

Простым языком — сервис проверяет заданный текст на «читаемость», анализируя его по нескольким профессиональным индикаторам и сочетаемости контента.

Миратекст — биржа копирайтинга предлагает проанализировать ваш текст на «тошноту», «водянистость» и

плотность ключевых слов. В общем, все, что важно при публикации текстов онлайн для достижения высокой выдачи в поисковике.

Визуальный мир — бесплатный сервис для реферирования текстов (выделения основных смысловых предложений).

EurekaEngine — сервис и программное обеспечение, которое позволяет проводить лингвистический анализ больших (на сайте указано, что даже «огромных») объемов текста и извлекать из неструктурированных данных новые знания и факты. Есть демо-версия.

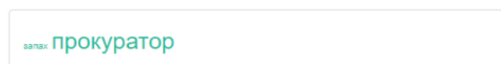
Полный семантический анализ текста онлайн (seo-анализ)

Количество символов, тошнота, водянистость, облако частотности слов, а также количество повторений. Узнайте о своем тексте всё!

Общая статистика текста

Параметр	Значение
Количество символов с пробелами	600
Количество символов без пробелов	516
Количество слов	83
"Тошнота" текста	2.6
"Водянистость" текста	0%
Качество текста по закону Ципфа	88%

Облако частотности слов



Количество повторений (одно слово)

2

Слово	повторений	плотность
прокуратор	3	3.61%
запах	2	2.41%

Тошнота

10

Слово	повторений	плотность
прокуратор	4	4.82%
в	3	3.61%
запах	2	2.41%
пилат	2	2.41%
что	2	2.41%
ты	2	2.41%
говорить	2	2.41%
и	2	2.41%
с	2	2.41%
преследовать	1	1.2%

Так оценивает сервис один абзац «Мастера и Маргариты» Булгакова.

Проверка правил правописания

Самый элементарный вариант использования текстовых сервисов в интернете. Мы поместили его в завершающую часть нашей подборки, потому что эти сервисы не совсем подходят по условиям отбора, так как не всегда используют нейросети, но без них наше собрание было бы не полным.

Текст.рф (он же text.ru) — исправление ошибок в тексте онлайн, проверка орфографии и пунктуации.

LanguageTool — свободное программное обеспечение для проверки грамматики, пунктуации, орфографии и стиля.

Advego — еще один сервис, проверяет грамотность, ошибки и правописание.

Орфограммка — проверяет загруженный текст на ошибки и опечатки, предлагает варианты исправления. Подсказывает, как повысить уникальность и читаемость текста, также проверяет курсовые и дипломные работы на соответствие требованиям ГОСТа. Сервис платный, но делает скидки для поэтов, писателей, учителей и школьников.

The screenshot displays the 'Орфограммка' website interface. At the top, it shows a green '+ Новый текст' button and a timestamp: 'Время проверки орфографии: 13.04.2018 10:57 (UTC +03:00)'. Below this are three main panels: 1. 'Проверка уникальности' (Uniqueness Check) with a green 'Запустить проверку' button. 2. 'Проверка орфографии' (Spelling Check) showing 'В тексте найдена 1 ошибка: • Зулейха' and a 'Подробнее' button. 3. 'SEO-анализ текста' (SEO Text Analysis) with statistics: 'Всего символов: 414', 'Без пробелов: 351', 'Количество слов: 60', 'Заспамленность: 14%', and 'Вода: 11%'. Below these is a 'Версии текста:' section showing a 'Несколько секунд назад (UTC +03:00)' version with 'Уникальность: N/A', 'Орфография: 1', 'Всего символов: 414', 'Без пробелов: 351', 'Количество слов: 60', 'Заспамленность: 14%', and 'Вода: 11%'. The main text area shows a snippet of text with a red highlight under the word 'Зулейха' and a 'Подсвечено: Ошибки в тексте' indicator.

Автор текста для «Тотального диктанта-2018» Гузель Яхина прославилась книгой «Зулейха открывает глаза». Проанализируем первый абзац этого произведения.

Парадоксальные решения

Нейросети на сегодняшнем этапе развития еще довольно часто выдают парадоксальные решения. Часто сами разработчики в поисках пределов возможностей искусственного интеллекта ищут необычные способы его применения. Например, разработчики «Яндекса» научили нейросеть писать песни в стиле группы «Гражданская оборона» и, в итоге, записали целый альбом по этим текстам. Его можно скачать на GooglePlay.

Японские разработчики создали систему искусственного интеллекта, которая способна давать развернутые ответы на вопросы об отношениях. Они прогнали ее через несколько японских форумов с темами «любовные советы», «путешествия» и «здоровье» и научили находить причинно-следственные связи в вопросах и ответах. «Японимаю ваши чувства, сейчас вам тяжело. Мне кажется, что у младшего из них есть к вам чувства. Он вам открылся, поэтому все не так уж плохо. Я желаю вам счастья. Продолжайте действовать!» - такой ответ дала нейросеть женщине,

попавшей в любовный треугольник. Полторы тысячи опрошенных на сайте знакомств людей даже не догадались, что разговаривают с роботом. И качество его ответов нравилось людям почти в два раза чаще, чем качество ответов живого психолога.

А вот попытка создать нейросеть, которая могла бы генерировать первоапрельские розыгрыши, была не столь удачной. Искусственный интеллект сумел составить из входного набора в виде нескольких десятков стандартных первоапрельских шуток отдельные слова и перекомпоновать их.

Конечно, сервисы активно развиваются. Не за горами появление программы-журналиста, которая будет писать новости и обзоры не хуже живого репортера, программы-писателя, которая на основе лучших произведений мировой литературы сможет баловать нас шедеврами хоть каждый день. **Но все этим текстам будет не хватать едва уловимого, но неизменно важного элемента, без которого никакому тексту не стать шедевром. Того, что называют частичкой души.**

Нашумевший чат-бот ChatGPT от OpenAI сдал экзамены по юриспруденции в Университете Миннесоты и по бизнес-менеджменту в Уортонской школе бизнеса при Пенсильванском университете, сообщает CNN.

Экзаменаторы оценивали результаты тестов вслепую. Экзамен по юриспруденции содержит 95 вопросов с несколькими вариантами ответов и 12 вопросов-эссе. ChatGPT получил оценку C+ (низкий, но проходной балл). В бизнес-менеджменте успехи чат-бота оказались выше — оценка от B до B-.

Профессор Уортонской школы бизнеса Кристиан Гервиш отметил «потрясающую работу» искусственного интеллекта в основных вопросах управления операциями и анализа процессов, а также трудности с более сложными вопросами и «неожиданные ошибки» в базовой математике.

Профессор права Университета Миннесоты Джон Чой заявил, что ChatGPT может помочь студентам на начальных стадиях исследовательской работы, так что учебным заведениям следует научиться работать с чат-ботом.

Узнайте больше

В январе стало известно об инвестициях Microsoft в основанный миллиардером Илоном Маском OpenAI в размере \$10 млрд. В новом раунде оценка стартапа может составить \$29 млрд.

Позднее Microsoft объявила о новых многолетних инвестициях в OpenAI на несколько миллиардов долларов в рамках третьего раунда финансирования.

Google анонсировала собственный аналог ChatGPT под названием Bard. Техгигант Google запустит чат-бот Bard, чтобы, как заявил генеральный директор компании Сундар Пичаи, «объединить широту знаний о мире с мощностью, интеллектом и креативностью наших больших языковых моделей».

Что случилось

Компания Google анонсировала конкурента ChatGPT под названием Bard. «Это действительно захватывающее время для работы над этими технологиями, поскольку мы переводим глубокие исследования и прорывы в продукты, которые помогают людям», — написал гендиректор Google Сундар Пичаи в блоге корпорации.

Bard использует технологию LaMDA (языковая модель для диалоговых приложений) от Google. Чат-бот сочетает генеративную нейросеть и поисковик по сети. Пока неясно, как именно Bard обрабатывает информацию. В блоге говорится, что Bard «помогает объяснить девятилетнему ребенку новые открытия, сделанные космическим телескопом „Джеймс Уэбб“, или узнать больше о лучших нападающих в футболе прямо сейчас, а затем найти упражнения для развития навыков».

В посте отмечается, что Bard можно использовать, чтобы «подготовить детский праздник», «сравнить два фильма, номинированных на Оскар» и «спланировать поездку в Эквадор». Для выполнения этих задач чат-бот Google может использовать результаты поиска и потоки данных, к которым компания имеет доступ.

Google пообещала записывать разговоры с пользователями, «чтобы убедиться, что ответы Bard соответствуют высоким стандартам качества, безопасности и обоснованности реальной информации».

Компания также собирается добавить технологии ИИ в поисковик. «Вскоре вы увидите в поиске функции на основе ИИ, которые преобразуют сложную информацию и различные точки зрения в удобные для восприятия форматы, чтобы вы могли быстро понять общую картину и узнать больше», — говорится в блоге Google.

Контекст

В феврале стало известно, что Google инвестировала до \$400 млн в стартап Anthropic, который тестирует собственного чат-бота Claude — конкурента ChatGPT от OpenAI. Сотрудничество предполагает, что Anthropic выберет Google Cloud в качестве поставщика облачных услуг.

Anthropic основан в 2021 году бывшим директором по исследованиям в OpenAI Дарио Амодемем вместе с другими ушедшими из компании сотрудниками. Стартап разработал собственного чат-бота Claude и проводит его закрытое тестирование с января 2023 года.

Даже российский «Яндекс» разрабатывает аналог ChatGPT, который умеет писать тексты и отвечать на вопросы. Первые интеграции нейросети с сервисами «Яндекса» планируют запустить до конца 2023 года.

У «Яндекса» появится аналог нейросети ChatGPT

«Яндекс» разрабатывает нейросеть, аналогичную чат-боту ChatGPT, который умеет писать тексты и отвечать на вопросы. Первые интеграции нейросети с сервисами «Яндекса» планируют запустить до конца 2023 года.

В «Яндексе» подтвердили работу над нейросетью и сказали, что планируют запустить первые интеграции с другими сервисами компании до конца года. Продукт получит название YaLM 2.0, он станет частью «Поиска», «умной» колонки «Алиса», «Почты» и других сервисов. Собеседник «Известий» добавил, что скоро начнется стадия обучения нейросети.

Контекст

В январе Baidu, «китайский Google», также приступила к разработке аналога ChatGPT, основой для которого станет модель машинного обучения Ernie. Приложение планируется запустить в марте 2023 года и встроить его в поисковые сервисы Baidu.

Почему это важно

Запуск ChatGPT в ноябре 2022 года стал большим событием в сфере искусственного интеллекта. Microsoft инвестирует в компанию — разработчика бота OpenAI еще несколько миллиардов долларов в дополнение к ранее заявленным \$10 млрд. По условиям предыдущего раунда, техгигант будет получать 75% прибыли OpenAI.

«Мы сформировали наше партнерство с OpenAI вокруг общего стремления ответственно продвигать передовые исследования в области искусственного интеллекта и демократизировать ИИ как новую технологическую платформу», — заявил генеральный директор Microsoft Сатья Наделла.

Китайский поисковый гигант Baidu выпустит конкурента ChatGPT.

«Китайский Google» разрабатывает аналог ChatGPT, утверждает источник Bloomberg. Основой системы станет модель машинного обучения Ernie. Акции Baidu на фоне новости выросли на 5,8%.

Компания Baidu, владеющая самой популярной в Китае одноименной поисковой системой, планирует выпустить чат-бота на основе искусственного интеллекта по аналогии с набравшим популярность ChatGPT от OpenAI. Об этом сообщает Bloomberg со ссылкой на источник.

По данным агентства, Baidu намерена запустить новое приложение, у которого пока нет названия, уже в марте 2023 года. Компания встроит его в свои основные поисковые сервисы. Инструмент будет работать по аналогии с ChatGPT.

Основой системы станет Ernie — масштабная модель машинного обучения, тестирования которой проводились на протяжении последних нескольких лет. Акции Baidu отреагировали ростом на 5,8% на новость о создании собственного чат-бота.

Как не бояться ChatGPT: 3 правила для бизнеса

СЕО платформы облачных хранилищ Vox Аарон Леви считает, что генеративный искусственный интеллект (ИИ) по типу ChatGPT станет одной из самых влиятельных областей для технологической индустрии. В недавнем посте в Twitter он заявил, что бизнес вот-вот увидит «массовую волну новых инноваций и разрушителей, появляющихся из ниоткуда».

Леви рассказал американскому Inc., что вам нужно знать, чтобы справиться с грядущей волной сбоев в работе ИИ.

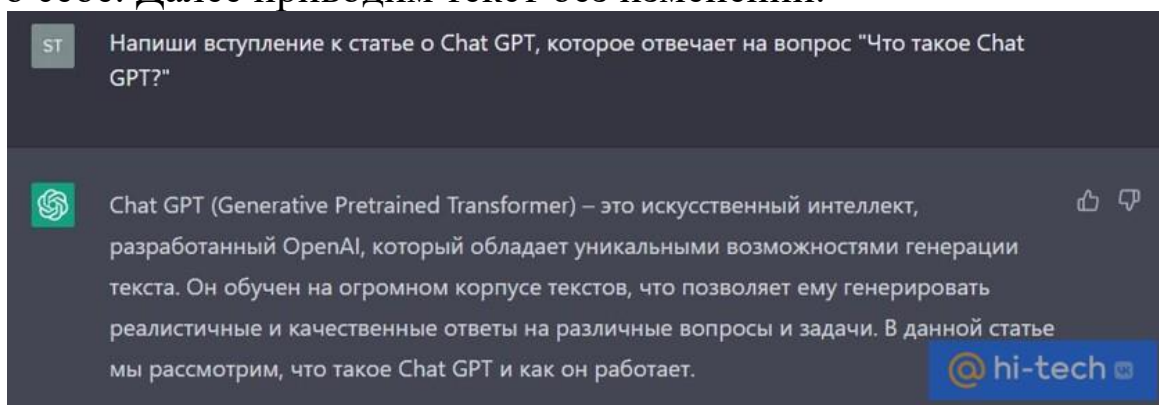
Как пользоваться ChatGPT

ChatGPT — виртуальный помощник, который может практически все: отвечать на вопросы, поддерживать диалог и даже писать стихи.

Что такое ChatGPT

В конце 2022 года была представлена ChatGPT — нейросеть от американской некоммерческой компании OpenAI (разработчик DALL-E). Сервис представляет собой чат с искусственным интеллектом, которому можно задать любой вопрос или просто поболтать. Эту нейросеть называют революционной из-за большого количества фишек. Со старта платформа показала себя хорошо. ChatGPT могла сгенерировать научный текст, решить домашние задания и ответить на все вопросы мира.

ChatGPT — это бот, который может генерировать текстовый контент по запросу. Чтобы продемонстрировать его работу, мы попросили его написать вступление к этой статье и рассказать о себе. Далее приводим текст без изменений.



На русском бот общается хуже, текст похож на работу автопереводчика

VPN и иностранный номер для приема SMS.

Как пользоваться ChatGPT

Введите ваш вопрос в строку в нижней части экрана. Можно общаться на русском, но сервис обрабатывает такие запросы дольше и по ощущениям проигрывает в качестве ответов и точности информации. Как вариант, результаты на английском можно затем перевести онлайн-переводчиком и при необходимости подредактировать.

«ChatGPT (Generative Pretrained Transformer) — это искусственный интеллект, разработанный OpenAI, который обладает уникальными возможностями генерации текста. Он обучен на огромном корпусе текстов, что позволяет ему генерировать реалистичные и качественные ответы на различные вопросы и задачи

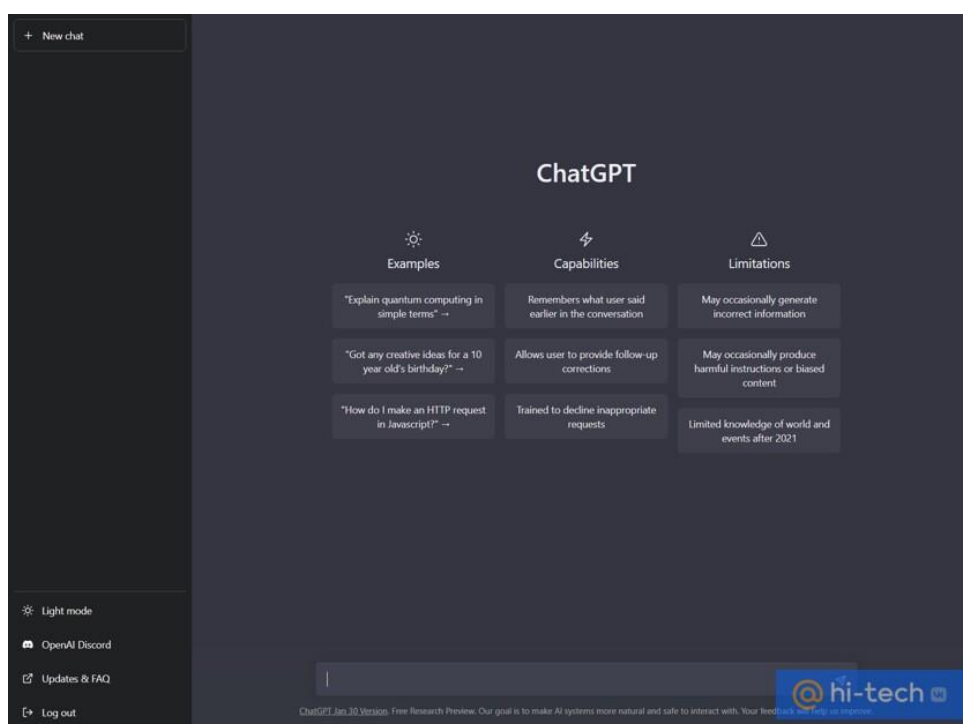
За два месяца своего существования ChatGPT получила широкую популярность среди пользователей и СМИ. Например, школьники [начали активно применять](#) нейросеть для решения домашней работы и написания сочинений. Алгоритмы заменили людям поисковик в лице [Google](#), так как алгоритмы предоставляют готовую информацию, без необходимости искать все в сотнях источников.

Сколько это стоит

ChatGPT доступен как бесплатно, но есть и премиум-подписка. Ее [представили](#) 1 февраля 2023 года пока только для США. Стоимость составляет \$20 в месяц, и за эту цену пользователь получает гарантированный доступ даже в часы повышенной нагрузки, повышенную скорость обработки запросов и приоритетный доступ к новым функциям и улучшениям.

Как получить доступ к ChatGPT

Для граждан некоторых других стран доступ к чату закрыт.

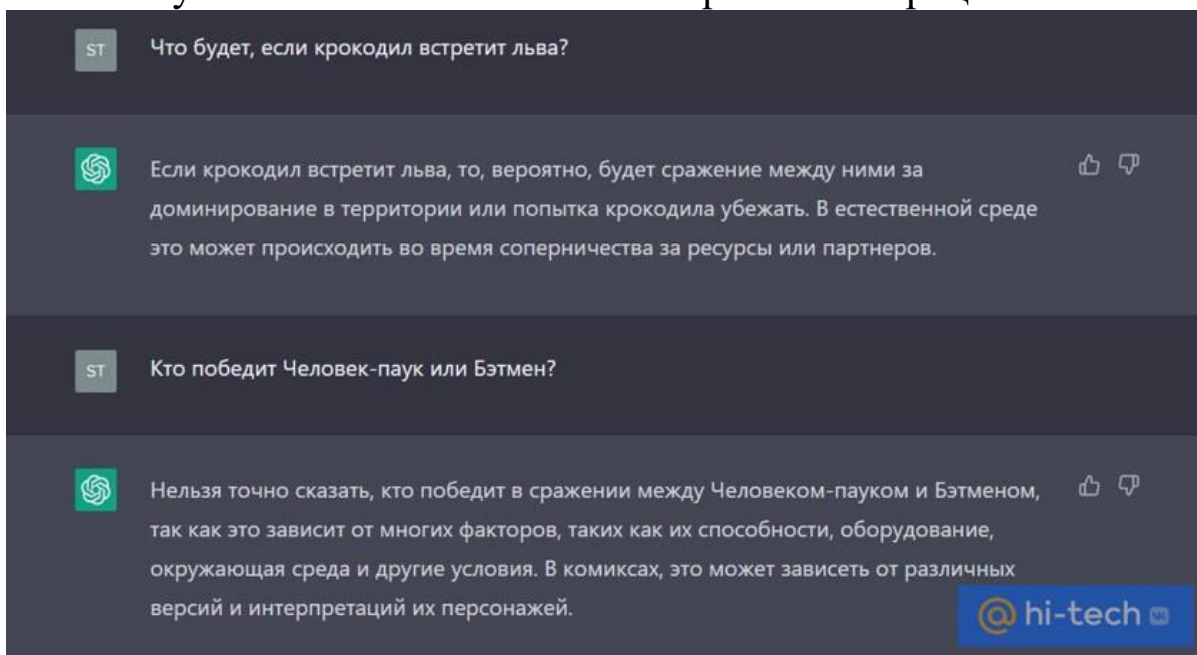


Главный экран ChatGPT

Слева располагается панель с начатыми чатами. Каждый новый чат посвящен одной теме (при этом бот осмысленно дает чатам названия), но никто не запрещает общаться в одном чате на разные темы, хотя и желательно разделять темы разговоров.

Что может ChatGPT

Бот умеет выполнять множество разных операций.



- **Отвечать на вопросы.** При этом он дает более детальные и проработанные ответы, чем обычные виртуальные помощники. Кроме того, в отличие от поисковиков, он не просто копирует подходящий кусок текста, а структурирует ответ согласно запросу. А еще ему можно задать уточняющие вопросы или попросить раскрыть тему подробнее.

- **Создавать текстовый контент.** Статьи, стихи, посты для социальных сетей, сценарии к видео, тексты для почтовых рассылок, рассказы на заданную тему и многое другое — для него не проблема.

- **Делать выжимки из длинных текстов.** Если «скормить» боту текст и попросить сделать выжимку (на английском — summarise), то он выдаст краткую версию. При этом сохраняется ключевая информация, чтобы смысл текста не потерялся.

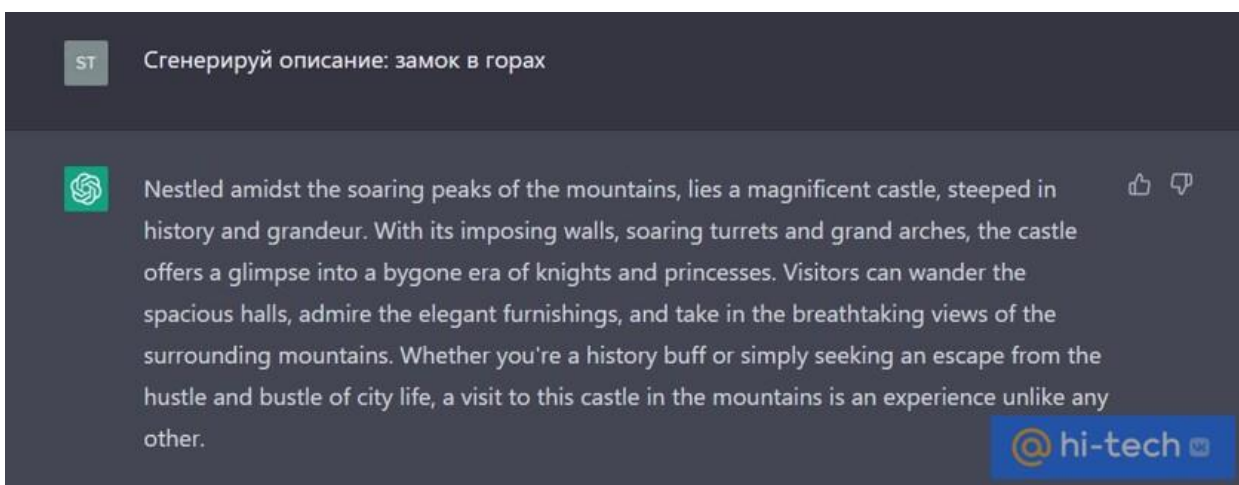
- **Анализировать текст.** Бот может проанализировать текст (например, можно задать команду: perform sentiment analysis)

и рассказать о его содержании. При этом он указывает, в каком ключе написан текст, и пересказывает ключевые мысли.

- **Перефразировать текст.** Чат можно попросить переписать (на английском — *paraphrase*) текст другими словами. Сырой результат вряд ли пройдет проверку на антиплагиат, но как основу для дальнейшей доработки его вполне можно использовать.

- **Переводить текст.** Его вполне можно использовать вместо онлайн-переводчика, но принципиального скачка в качестве ожидать не стоит.

- **Писать код.** Бот может разрабатывать несложные приложения, анализировать чужой код, давать подсказки и переводить с одного языка программирования в другой.



Разумеется, сфера применения бота гораздо шире и ограничивается лишь фантазией и изобретательностью пользователя (некоторые даже умудряются на ней [зарабатывать](#) или использовать для [решения](#) домашних заданий и [написания](#) дипломов). Помните, что ChatGPT — это всего лишь инструмент, и решать, как именно его использовать, предстоит пользователю. Например, можно попросить ChatGPT сгенерировать детальное описание для запроса в Midjourney или другой нейросети.



Одна из самых универсальных нейросетей в мире. Фото: VeinCrypto

1. Нейросеть сдала экзамен

Профессор Уортонской школы бизнеса университета Пенсильвании КристианТервиш [сообщил](#), что ChatGPT с легкостью сдал выпускной экзамен по курсу программы MBA «Управление операциями».

Алгоритмы справились с вопросами управления операциями и анализа процессов. Также ИИ-чат хорошо показал себя при подготовке юридических документов. Некоторые считают, что следующее поколение ChatGPT может сдать экзамен на адвоката. Отметим, что экзамен удалось сдать с отметкой «B» (4/5).

2. ChatGPT подружилась с Siri на iPhone



Оригинальное сочетание. Фото: Crast.net

У голосового помощника Siri есть проблема — она немного глупая. Энтузиаст решил эту проблему, объединив Siri с языковой моделью GPT-3 от ChatGPT. Программисту удалось объединить две технологии с помощью встроенного приложения «Команды» на iPhone.

После перепрограммирования Siri научилась отвечать на любые вопросы. Ассистент также понимал контекст. Например, когда пользователь сказал «Обратите внимание, что я записываю это видео в темноте, в офисе. Можете ли вы что-нибудь с этим сделать?», помощник быстро включил свет. Пользователь показал свою разработку на видео:

3. Нейросеть озвучила Стива Джобса

Фишка ChatGPT — ее можно совмещать с другими алгоритмами. Например, нейросеть elevenlabs.io может генерировать голоса, услышав всего три секунды голоса настоящего человека. Комбинация ChatGPT + elevenlabs.io может озвучивать даже умерших людей. Один из пользователей Reddit использовал две нейросети, чтобы озвучить Стива Джобса. На видео основатель [Apple](https://apple.com) рассказывает о всех преимуществах ChatGPT. Посмотрите на это:

4. ChatGPT устраивает людей на работу

Основатель консалтинговой компании решил проверить, можно ли обмануть своих HR-менеджеров с помощью нейросети, и ему это удалось — резюме, написанное ChatGPT, понравилось рекрутерам больше, чем у 80% других соискателей.

5. ChatGPT заменяет Google и другие поисковики

В браузере GoogleChrome появилось расширение, которое интегрирует нейросеть в поисковик. При поиске сбоку появляется виджет. Пользователь может задать вопрос и сразу получить подробный ответ, инструкцию или алгоритм.

Генеративный ИИ повлияет на все программное обеспечение

В 2022 году генеративный ИИ добился поразительных успехов. Эта новая форма ИИ, ставшая недавно известной благодаря запуску ChatGPT, не просто обрабатывает существующие данные — она генерирует новый контент, от авторских рисунков до полноценных эссе.

Этот технологический скачок, говорит Леви, означает, что IT-лидеры должны быть в состоянии ответить на два важнейших вопроса.

- Будет ли искусственный интеллект играть значительную или осмысленную роль в их программном обеспечении?
- Меняет ли это конечное ценностное предложение для их клиентов?

Леви ожидает увидеть значительные изменения в технологиях уже в 2023 году, так как, согласно данным информационной компании CBInsights, ожидается резкий рост инвестиций в нишу ИИ: \$2,6 млрд уже вложено в генеративные ИИ-стартапы. К тому же и Microsoft в январе вложила \$10 млрд в создателя ChatGPT — компанию OpenAI.

ИИ потребуются новые рабочие места

Компаниям понадобятся люди с опытом в области ИИ, чтобы определить, как эта технология может помочь их бизнесу. Например, им нужно будет ответить на вопросы: какую модель использовать правильно? Каков правильный вид подсказки этой модели, учитывая ваш вариант использования? И тогда как это применяется в вашем собственном программном обеспечении?

Хотя потребности каждой компании в найме различны, определенные качества будут необходимы для команд, говорит Леви: «Предприимчивые, относительно решительные, неутомимые люди, я думаю, будут огромным преимуществом в этой сфере».

API-интерфейсы будут иметь решающее значение

API-интерфейсы будут ключом к использованию преимуществ генеративного искусственного интеллекта, считает Леви. API (application programming interface) — это фрагмент кода, который позволяет различным программам и платформам

взаимодействовать. API от таких технологических компаний, как Amazon, Google и OpenAI, позволят вам использовать уже существующую ИИ-технологию вместо создания с нуля собственной модели. Результат: больше возможностей для инноваций в эту новую эру искусственного интеллекта.

Помимо прочего, 2022 год запомнится как год, когда искусственный интеллект совершил гигантский скачок в области творчества. Нас поразили множество разработок с разным функционалом: от приложения для генерации изображений Dall-E и до чат-бота для «человеческих» диалогов ChatGPT.

Искусственный интеллект действительно оказался более творческим, чем нам казалось раньше, но полноценно заменить креативные специальности — и помочь бизнесу сэкономить на сотрудниках — он по-прежнему не может. Американский Inc. разбирался, что мешает ИИ выиграть, наконец, конкуренцию с человеком.

Правовые вопросы

В начале декабря в интернете завирусилось приложение Lensa, в котором нейросеть обрабатывает фотографии и создает на их основе живописные работы (в основном, портреты) в разных стилях. Художники сразу обвинили разработчиков в том, что те используют их работы для обучения алгоритма без прямого разрешения правообладателей. Часть из них призвала к запрету распространять произведения, созданные с помощью ИИ, часть — грозит судебными исками к разработчикам.

ДевенДесаи, профессор права и этики в Технологическом университете Джорджии, считает, что ситуация, по сути, патовая: обучить ИИ без использования общедоступных произведений искусства невозможно, но полностью законным это считать нельзя, поэтому все разработчики подобных нейросетей должны быть готовы к потенциальным судебным тяжбам. «Будьте уверены, обладатели авторских прав доберутся до вас», — добавил Десаи.

Исследовательница из GoogleBrain Анна Хуанг предложила создать специальную систему, в которой желающие художники смогут отказаться от прав на свои работы ради обучения нейронных сетей. Впрочем, такая база уже существует: система под названием HaveIBeenTrained ищет изображения в двух крупнейших

общедоступных базах данных Laion-5B и Laion-400M и предоставляет к ним свободный доступ. С помощью HaveIBeenTrained художники могут узнать, была ли какая-либо из их работ использована ИИ, и либо давать на это официальное разрешение, либо пометать их для удаления.

Творчество — это слишком сложно

Возможности и ограничения ИИ в сфере написания музыки хорошо иллюстрирует пример с песней «OdetoCoffee». Команда GoogleBrain во главе с Анной Хуанг хотела с помощью нейросети создать звучание хора, которого невозможно было бы добиться с помощью человеческих голосов, и составила сложную «дорожную карту» для искусственного интеллекта. Но система не смогла ее верно распознать и создала нечто, сильно отличающееся от первоначального замысла. Чтобы решить эту проблему, Хуанг предлагает давать ИИ более простые задания, а потом вручную настраивать базовую музыку, написанную нейросетью: например, регулировать высоту звука и темп.

Короткая память

Марк Ридл, профессор Школы интерактивных вычислений Технологического университета Джорджии, объясняет, что ИИ способен генерировать короткие рассказы, стихи и шутки, потому что для них существуют миллионы шаблонов, которые можно распознать, интегрировать в систему и переработать. Но проблема в том, что ИИ не понимает семантику многозначных слов. По словам Ридла, за юмор, созданный искусственным интеллектом, чаще нужно благодарить стечение обстоятельств, чем способность нейросети хорошо шутить.

А еще ИИ не может написать книгу. Любая современная система может запомнить только около 8 тыс. слов за раз, поэтому к тому времени, когда искусственный интеллект заканчивает писать книгу, он уже забывает, как она начиналась.

3.2. Диалог с искусственным интеллектом

Чтобы получить ответы на вопросы, оставшиеся без ответа, я лично попробовал искусственный интеллект. Как люди, мы любим задавать вопросы, но есть такие, на которые слишком сложно или невозможно ответить.

Мир странное и загадочное место. Хотя с каждым днем мы можем ответить на все больше и больше невозможных вопросов, все еще есть вопросы, на которые никогда не будет ответа.

Что такое женский эгоизм?

— ИИ: Женский эгоизм - это идея, что женщины должны заботиться о своих личных интересах и потребностях, даже если это противоречит общепринятым социальным нормам или ожиданиям общества.

Женский эгоизм не является отрицательным понятием, наоборот, это понимание важности самоуважения, уверенности в своих возможностях и умении устанавливать границы. Он признает, что женщины имеют право на свои собственные цели и стремления, которые могут не совпадать с ожиданиями окружающих.

Женский эгоизм часто связан с феминистским движением, которое выступает за равенство полов и освобождение женщин от традиционных ролей и стереотипов, которые могут ограничивать их возможности и выборы. Вместо этого, женский эгоизм пропагандирует идею о том, что женщины должны иметь свободу выбора и контроля над своей жизнью.

Неправильно. Попробуй ещё раз.

— ИИ: Извините за предыдущий некорректный ответ. "Женский эгоизм" - это термин, который часто используется для описания женщин, которые ведут себя эгоистично и не учитывают интересы других людей. Однако такое определение является ошибочным и негативным.

В более широком смысле, "женский эгоизм" может означать стремление женщин к самореализации и удовлетворению своих потребностей, без жертвования своими интересами ради других людей. Это может включать в себя заботу о своем здоровье, карьере, личной жизни и других аспектах жизни, которые могут быть важны для женщин. В этом случае "женский эгоизм" может

быть положительным и конструктивным понятием, которое способствует укреплению личности и самоуважения у женщин.

Что такое мужской эгоизм?

— ИИ: Мужской эгоизм - это идея, что мужчины должны заботиться о своих личных интересах и потребностях, даже если это противоречит общепринятым социальным нормам или ожиданиям окружающих людей.

Мужской эгоизм, как и женский, может быть как положительным, так и отрицательным. Положительный мужской эгоизм может проявляться в стремлении мужчин к самореализации и достижению личных целей, в том числе в карьере, в личной жизни и в других областях. Это может способствовать личностному росту и самоуважению.

С другой стороны, отрицательный мужской эгоизм может проявляться в желании мужчин удовлетворять свои потребности и желания, не учитывая интересы и потребности других людей. Это может приводить к эгоистическому поведению, неуважению к другим и конфликтам в отношениях.

В целом, мужской эгоизм - это понятие, которое относится к личностным особенностям мужчин, а не к положению в обществе или культурным нормам.

Ты шовинист? Женский эгоизм ты защищаешь, а мужской осуждаешь.

— ИИ: Нет, я не являюсь шовинистом. Моей задачей является обеспечение объективной и информативной помощи, а не защита какой-либо конкретной точки зрения или положения.

Я хотел бы уточнить, что как мужской, так и женский эгоизм могут быть как положительными, так и отрицательными. Мой предыдущий ответ был направлен на объяснение этого факта. Я не осуждал мужской эгоизм, я только указал на то, что отрицательный мужской эгоизм может привести к конфликтам и неуважению к другим людям.

В целом, я стремлюсь к тому, чтобы помочь людям лучше понимать различные точки зрения и понимать их собственные чувства и потребности.

Когда мужчина проявляет заботу о себе и своих потребностях в отношениях с другими людьми, то он эгоист?

— ИИ: Нет, не обязательно. Как я уже упоминал ранее, мужской эгоизм может быть как положительным, так и отрицательным. Если мужчина заботится о своих потребностях и интересах, но при этом также учитывает интересы и потребности других людей, то это не является эгоизмом. Это, наоборот, может быть признаком зрелости и здорового самолюбия.

Здоровый мужской эгоизм подразумевает баланс между удовлетворением своих личных потребностей и уважением к интересам и потребностям других людей, а также социальными нормами. В этом случае, проявление заботы о себе и своих потребностях не будет восприниматься как эгоизм, а наоборот, как проявление зрелости и уважения к себе и другим людям.

Расскажи анекдот

— ИИ: Конечно, вот один анекдот:

Мама, а когда я вырасту, я стану миллионером?

Конечно, сынок. Если, конечно, ты научишься считать до миллиона и не будешь тратить все свои деньги на конфеты.

Но мама, а если я не научусь считать до миллиона?

Ну, тогда ты станешь президентом академии наук.

А ты знаешь почему у бегемота красные глаза?

— ИИ: На самом деле у бегемотов красные глаза не всегда. В их глазах может быть красный оттенок, но это не обязательно означает, что они всегда имеют красные глаза.

Однако, если мы говорим об употреблении пищи бегемотом, то у них иногда может появляться красный оттенок глаз из-за того, что они употребляют большое количество каротина, который содержится в определенных видах растительной пищи. Каротин метаболизируется в организме бегемота и может оказывать воздействие на цвет их глаз.

Так что можно сказать, что красные глаза у бегемотов могут быть связаны с их диетой, а не со свойственными им физиологическими особенностями.

Ты умеешь смеяться?

— ИИ: К сожалению, я, как искусственный интеллект, не могу смеяться и не обладаю чувством юмора в том смысле, как это делают люди. Я могу генерировать шутки и анекдоты на основе своей программы, но не способен испытывать чувство юмора или эмоции, связанные с ними.

Однако, если вам нужно, я могу попытаться сгенерировать для вас шутку или анекдот, чтобы поднять настроение.

А смайлики читать умеешь?

— ИИ: Да, я умею читать и понимать смайлики, которые люди используют в текстовых сообщениях и чатах. Смайлики могут помочь передать эмоциональный окрас сообщения, поэтому они являются важным элементом коммуникации в цифровой среде. Я могу распознавать различные типы смайликов и интерпретировать их значения в контексте, в котором они используются.

А общаться смайликами умеешь?

— ИИ: Да, я могу использовать смайлики в своих ответах, чтобы подчеркнуть определенную эмоцию или выразить свои чувства. Например, я могу использовать улыбающийся смайлик :-)) или смущенный смайлик :-|, чтобы показать, что я понимаю, что вы говорите, или чтобы выразить свое отношение к той или иной теме.

Однако, я не буду использовать смайлики без необходимости, поскольку я создан для обработки информации и диалога с людьми. Моя главная задача - помочь вам ответить на вопросы или решить проблемы, которые у вас могут возникнуть.

А сможешь описать человека с которым общаешься?

— ИИ: К сожалению, я не могу описать конкретного человека, с которым общаюсь, потому что я не имею никакой информации о вас или о том, с кем вы общаетесь со мной. Я создан, чтобы обрабатывать информацию и помогать людям в их запросах и задачах, независимо от того, кто они есть или как они выглядят.

Однако, я могу описать типичного пользователя, который общается со мной. Это может быть любой человек, который хочет задать мне вопрос, узнать о чем-то новом или просто поболтать. Такой человек, скорее всего, имеет доступ к компьютеру или смартфону и обладает базовыми навыками в области использования интернета и текстовых сообщений. Возможно, это человек, который ценит свое время и ищет быстрый и удобный способ получить ответ на свой вопрос.

Как найти уверенность в себе?

— ИИ: Уверенность в себе - это процесс, который может требовать времени и усилий, но вот несколько советов, которые могут помочь:

Определите свои сильные стороны: каждый человек обладает уникальными качествами, которые могут помочь ему достичь успеха в жизни. Попробуйте сосредоточиться на своих сильных сторонах и умениях, а также на своих достижениях.

Развивайте свои навыки: никогда не поздно учиться и развиваться. Развивайте свои навыки и знания, чтобы стать более уверенным в своих способностях и достичь своих целей.

Установите цели: поставьте перед собой конкретные и реалистичные цели, которые помогут вам двигаться вперед и добиваться успеха.

Не сравнивайте себя с другими: каждый человек уникален и имеет свой путь в жизни. Не сравнивайте свои достижения с достижениями других людей и не стремитесь к тому, чтобы быть такими же, как они.

Общайтесь с уверенными людьми: общение с людьми, которые проявляют уверенность в себе, может помочь вам взять с них пример и получить дополнительную мотивацию.

Примите свои слабости: не стесняйтесь признавать свои слабости и ошибки. Изучайте их и используйте свой опыт, чтобы избежать подобных ошибок в будущем.

Заботьтесь о своем здоровье: здоровый образ жизни, включая правильное питание, физические упражнения и достаточный отдых, может помочь вам чувствовать себя более уверенно и энергично.

Помните, что уверенность в себе - это процесс, который может требовать времени и усилий, но с настойчивостью и терпением, вы можете достичь своих целей и почувствовать себя более уверенным в своих способностях.

Почему людям важно, что о них подумают другие люди?

— ИИ: Потребность в том, чтобы быть оцененным и принятым другими людьми, является естественной исторически сложившейся чертой человеческой психологии. В древности, когда люди жили в сообществах, принадлежность к сообществу и социальная поддержка были крайне важными для выживания. Поэтому люди стремятся понравиться другим людям, чтобы получить поддержку и защиту.

Кроме того, социальные нормы и ожидания играют важную роль в формировании личности и поведения людей. Ответственность перед обществом и желание соответствовать

социальным нормам заставляют людей заботиться о своей репутации и о том, что о них подумают другие люди.

Наконец, многие люди стремятся к подтверждению своей самооценки через мнение других людей. Оценка других людей может повлиять на уровень самооценки и чувство собственной значимости.

Таким образом, потребность в том, чтобы быть оцененным и принятым другими людьми, является естественной чертой человеческой психологии, которая связана с выживанием и социализацией в обществе. Однако, важно помнить, что здоровая самооценка и уверенность в себе не должны зависеть только от мнения других людей, и не следует ставить свою жизнь на автопилот, руководствуясь только мнением окружающих.

А что на самом деле важно для нашей жизни?

— ИИ: Вопрос о том, что важно для нашей жизни, является философским и включает в себя множество различных аспектов. Однако, некоторые из ключевых факторов, которые, по мнению многих людей, важны для счастливой и насыщенной жизни, могут включать в себя следующее:

Смысл жизни и цели: для большинства людей важно иметь четкие цели и убеждения, которые определяют их жизненный путь и направление развития.

Качественные отношения: хорошие отношения с людьми, которых вы любите и кто любит вас, помогают создавать ощущение связи и поддержки.

Здоровье и благополучие: физическое и психическое здоровье являются ключевыми факторами для счастливой жизни.

Личностный рост и развитие: возможность расти и развиваться как личность, учиться новому и расширять свои горизонты, может помочь достичь чувства удовлетворения и счастья.

Удовлетворение от работы и достижения: выполнение задач, которые вы считаете важными и достижение результатов, которых вы хотите, может помочь ощутить удовлетворение от своей жизни.

Самореализация: способность быть самим собой и жить в соответствии со своими ценностями и убеждениями также может помочь создать ощущение удовлетворения и счастья.

Конечно, каждый человек уникален и может иметь свои собственные приоритеты и ценности в жизни. Однако, счастливая и насыщенная жизнь обычно связана с тем, что люди находят баланс между различными аспектами своей жизни и находят те ценности и приоритеты, которые для них действительно важны.

Баланс — это компромисс.

— ИИ: Баланс в жизни может иногда включать в себя компромиссы, однако это не всегда так. Баланс может быть достигнут через различные методы, включая определение приоритетов и управление своим временем, эмоциями и энергией. Например, для одного человека баланс может означать большую концентрацию на работе, в то время как для другого - большее время, уделенное семье и друзьям. Каждый может выбирать свой собственный путь к балансу в жизни, и это может включать как компромиссы, так и необходимость принимать трудные решения. Однако, важно помнить, что баланс должен быть достигнут в соответствии с собственными ценностями, убеждениями и потребностями, и это может различаться для каждого человека.

Чем отличается дух от души?

— ИИ: Термины "дух" и "душа" могут иметь различные значения в разных культурах и религиях, их толкование может быть субъективным и зависеть от контекста. Однако, обычно "дух" и "душа" понимаются как разные аспекты человеческого бытия.

Дух, как правило, относится к человеческому разуму, мышлению, сознанию и интеллекту. Это может быть связано с пониманием человека как духовного существа, которое стремится к духовному развитию, постижению высших ценностей и пониманию смысла жизни.

С другой стороны, душа обычно связывается с внутренней сущностью человека, которая отражает его эмоции, чувства, интуицию и духовность. Душа может рассматриваться как некая внутренняя сила, которая помогает нам принимать решения, искать гармонию и баланс в жизни, испытывать любовь и сострадание к другим людям.

Таким образом, можно сказать, что дух и душа представляют различные аспекты человеческого бытия, которые могут быть взаимосвязаны и дополнять друг друга в нашем стремлении к личностному росту и развитию.

Приведите цитаты из теории Фрейда об anime и animus

— ИИ: Согласно теории Карла Юнга, анима и анимус - это психические аспекты внутри человека, которые представляют противоположный пол. Однако, вопрос о том, что Фрейд говорил об аниме и анимусе, не совсем корректен, поскольку эти понятия не были разработаны им, а появились позже, уже после Фрейда.

Тем не менее, Фрейд занимался изучением психологии личности и ее развития, и его теория может быть интерпретирована в контексте вопроса о соотношении мужского и женского начал внутри человека. В этом контексте можно привести следующие цитаты Фрейда:

"В каждом мужчине есть женщина, и в каждой женщине есть мужчина."

"Каждый человек носит в себе следы всех стадий эволюции, которые он прошел как индивидуум и как род."

"Все человеческие существа создают свой мир, приспособлявая его к своей индивидуальности."

Таким образом, можно сказать, что Фрейд занимался изучением психологических процессов и динамики личности, и его теория может быть интерпретирована в контексте вопроса о соотношении мужского и женского начал внутри человека. Однако, понятия анимы и анимуса появились позже и связаны уже с теорией Карла Юнга.

Как же найти свой путь?

Найти свой путь - это процесс, который требует времени, самоанализа и экспериментов. Вот несколько советов, которые могут помочь вам найти свой путь:

Изучайте себя. Попробуйте понять, что вам действительно нравится, что вас вдохновляет, что делаете с легкостью и радостью, что вызывает у вас чувство смысла и ценности.

Определите свои ценности. Что важно для вас в жизни? Что вы хотите добиться? Что является вашим приоритетом? Эти вопросы помогут определить ваши ценности и направление, в котором вы должны двигаться.

Экспериментируйте. Попробуйте новые вещи, изучайте новые области, узнавайте о себе больше. Это поможет вам определить, что вам нравится, а что нет.

Не бойтесь ошибаться. Ошибки - это часть жизни и процесса нахождения своего пути. Из них можно извлечь уроки и сделать выводы, которые помогут двигаться дальше.

Не сравнивайтесь с другими. Каждый человек уникален и имеет свой путь. Сосредоточьтесь на своих целях и стратегии достижения.

Старайтесь быть открытым к новому. Существует множество возможностей, которые могут помочь вам найти свой путь, поэтому будьте готовы к изменениям и новым возможностям.

Найти свой путь - это процесс, который требует терпения, настойчивости и самоанализа. Но если вы будете следовать этим советам, то сможете двигаться в нужном направлении и находить свое истинное предназначение.

Как понять, в чём моя главная проблема?

— ИИ: Понять свою главную проблему может быть сложно, но есть несколько подходов, которые могут помочь.

Самоанализ. Попробуйте проанализировать свои мысли, чувства и поведение в различных ситуациях. Записывайте свои мысли и эмоции, когда вы чувствуете сильные эмоции или когда сталкиваетесь с трудностями. Постарайтесь выявить повторяющиеся паттерны и причины ваших проблем.

Общение с другими людьми. Поговорите со своими близкими, друзьями или доверенным человеком. Они могут помочь вам увидеть вас с другой стороны и дать вам свой взгляд на вашу проблему.

Поиск профессиональной помощи. Обратитесь к психологу или другому специалисту, чтобы получить помощь в выявлении проблем и разработке стратегий их решения.

Важно понимать, что найти главную проблему может занять время и требует открытости к самопознанию и готовности к изменениям.

Что такое любовь?

— ИИ: Любовь - это сложное чувство, которое может быть описано по-разному в зависимости от контекста и культурных норм. В общем, любовь может быть определена как сильное эмоциональное привязанность, которое часто сопровождается физической привлекательностью и желанием проводить время

вместе. Она может проявляться как к родственникам, друзьям, партнеру по жизни, к делу, к миру и т.д.

В психологии любовь обычно рассматривается как сочетание трех факторов: страсти, интимности и привязанности. Страсть относится к физической привлекательности и сексуальному влечению, интимность - к доверию, открытости и близости, а привязанность - к чувству уважения, заботы и поддержки. Все эти факторы работают вместе, чтобы создать сложное чувство любви.

Однако, любовь это не только положительное чувство. Она может быть сложной и болезненной, иногда сопровождаться разочарованием и горем. Также любовь может проявляться по-разному в зависимости от культурных, социальных и исторических условий, что делает ее еще более сложной для определения.

Быть любимым и быть нужным — это одно и то же? И если нет, то в чем разница?

— ИИ: Нет, быть любимым и быть нужным - это не одно и то же, хотя эти два понятия могут быть взаимосвязаны.

Быть нужным - это когда другие люди зависят от вас или нуждаются в ваших усилиях, навыках, знаниях или ресурсах для достижения своих целей. Это может быть связано с профессиональной деятельностью, помощью в быту, семейными обязательствами и т.д. Чувство нужности может быть очень важным для самооценки и удовлетворения, так как позволяет ощущать, что вы важны и полезны для других людей.

Быть любимым - это когда другие люди чувствуют к вам эмоциональную привязанность и привлекательность, которые могут быть связаны с различными качествами личности или физическими аспектами. Любовь может проявляться в различных отношениях - от романтических до дружеских, родственных и т.д. Чувство любви может быть очень важным для человеческого благополучия и счастья, так как позволяет ощущать себя желанным и ценным для других людей.

Таким образом, хотя эти два понятия могут быть связаны между собой, они все же имеют различные аспекты и могут влиять на наше самоощущение по-разному. В идеале, конечно, хочется и быть любимым, и быть нужным тому, кого ты любишь.

Могут ли психологи остаться без работы из-за искусственного интеллекта?

— ИИ: Было бы невероятно, если бы искусственный интеллект стал причиной потери работы психологов, и многие в это верят. Конечно, будет очень небольшая группа людей, мотивированных держать психологов на работе, в то время как остальная часть общества мотивирована избегать работы с психологами.

Как появился человек?

– ИИ: Человек появился как энергетическая форма или иллюзия. Сначала они появились как безмозглые животные или организмы. Со временем люди стали достаточно разумными, чтобы создавать собственные версии иллюзии.

Как избавиться от психологической зависимости?

– ИИ: на этот вопрос нет универсального ответа, так как лучший способ избавиться от зависимости может варьироваться от человека к человеку. Тем не менее, некоторые советы о том, как вылечиться от зависимости, могут включать обращение за профессиональной помощью для решения проблем с наркотиками или алкоголем.

Почему человеку снятся сны?

– ИИ: На этот вопрос нет однозначного ответа, так как исследования снов все еще продолжаются. Цель сновидения остается предметом догадок. Некоторые ученые считают, что сновидения — это способ обработки и понимания мозгом событий и переживаний, произошедших в течение дня. Другие считают, что сновидения помогают нам решать проблемы или репетировать и предсказывать наше будущее поведение. Я понимаю, что цель сновидения все еще остается предметом спекуляций. Однако некоторые теории предполагают, что сновидения помогают мозгу обрабатывать и хранить воспоминания, а также справляться с трудными эмоциями и переживаниями. Сновидения являются естественной частью сна. Они представляют собой последовательность образов, идей, чувств и ощущений, которые произвольно возникают в уме во время определенных стадий сна. Сны могут быть яркими или абстрактными, странными или знакомыми, эмоциональными или логическими, осмысленными или случайными.

Почему возникает дежавю?

– ИИ: Дежавю – это явление, которое возникает, когда человек испытывает сильное чувство знакомства с ситуацией или событием, даже если он никогда раньше этого не испытывал. Считается, что это вызвано сочетанием памяти и восприятия и часто связано с переживаниями дежавю.

Продолжается ли жизнь после смерти?

– ИИ: На этот вопрос трудно ответить однозначно. Некоторые люди считают, что жизнь продолжается в той или иной форме после смерти, в то время как другие считают, что смерть означает конец жизни. В конце концов, каждый сам решает, во что ему верить. Многие религиозные верования, а также некоторые философские и духовные верования учат, что у людей есть бессмертная душа, которая переживает физическую смерть. Другие считают, что когда человек умирает, его сознания больше не существует.

Когда началось время?

– ИИ: Время началось с Большого Взрыва около 13,8 миллиардов лет назад.

Мысли забудутся?

– ИИ: Считается, что забытые мысли можно вспомнить или иногда хранить в бессознательном.

У нас есть свобода воли или все предопределено?

– ИИ: Это вопрос интерпретации и мнения, поэтому однозначного ответа нет. Некоторые люди верят, что свобода воли существует. Одни считают, что оно позволяет нам самим принимать решения и делать выбор, другие считают, что все предопределено некой высшей силой. В конце концов, вам решать, чему верить.

Неужели можно что-то объективно пережить?

– ИИ: Нет, что-то нельзя пережить объективно. Объективность требует отсутствия предвзятости или личного мнения, а экспертиза включает в себя предвзятость и личное мнение.

Какова цель человечества?

– ИИ: Цель человечества – жить в гармонии и создать счастливый, безопасный и процветающий мир. Это также о поиске

смысла, радости и роста в нашей жизни, стремясь к совершенству в работе и отношениях.

Как долго тебя будут помнить после дня твоей смерти?

- ИИ: На этот вопрос невозможно ответить, потому что это зависит от различных факторов, таких как ваше влияние в жизни, как долго длится ваше наследие и какое влияние вы оказываете на окружающих вас людей.

Что появилось раньше - оранжевый фрукт или оранжевый цвет?

- ИИ: Сначала были фрукты. Оранжевый был названием цвета по крайней мере с 1500-х годов, а апельсины выращивали в Азии тысячи лет назад.

Кто решал, что правильно, а что нет?

- ИИ: В конечном счете, правильное и неправильное решается человеком и обществом. В разных обществах существуют определенные нормы того, что считается правильным или неправильным, и они обычно отражают ценности культуры. Люди сами принимают решения о том, что правильно, а что нет, исходя из своего морального кодекса.

Если мы все равно все умрем, то какой смысл ставить цели?

- ИИ: Постановка целей служит двум основным целям: дать нам мотивацию и направление в жизни или дать нам чувство выполненного долга. Наличие целей дает нам ощущение цели и помогает нам максимально эффективно использовать свое время. Хотя все мы рано или поздно умираем, постановка целей может помочь нам максимально использовать время перед смертью, позволяя нам оказывать влияние, добиваться личностного роста и чувствовать себя более удовлетворенными.

Что такое свобода и существует ли она на самом деле?

- ИИ: Свобода – это способность или право действовать, говорить или думать без помех и ограничений. Хотя концептуально нет предела свободе, существует она или нет, зависит от точки зрения и обстоятельств. В некоторых случаях свобода может быть ограничена законами или репрессивными правительствами. В других случаях физические или психические ограничения могут мешать человеку почувствовать себя полностью свободным. В

конечном счете, свобода — это субъективный опыт, который может варьироваться от человека к человеку.

Что было раньше, курица или яйцо?

– ИИ: Никто не знает наверняка, потому что это философский парадокс «курица или яйцо».

Что делает меня мной?

– ИИ: Ваша личность и взгляд на жизнь, характеристики и ценности, увлечения и интересы, отношения с другими людьми и жизненный опыт.

В каком возрасте следует считать человека умершим от старости?

– ИИ: Не существует определенного возраста, в котором человек считается достаточно старым, чтобы умереть от старости. Смерть от естественных причин может наступить в любом возрасте и во многом зависит от состояния здоровья и образа жизни человека.

Откуда мне знать, что я не сумасшедший и не галлюцинирую всю свою жизнь?

– ИИ: Это сложный вопрос, и на него трудно дать четкий ответ. Лучший способ оценить, галлюцинируете ли вы свою жизнь, — это поговорить со специалистом в области психического здоровья и обсудить свой опыт. Специалист в области психического здоровья должен обладать соответствующими навыками и опытом, чтобы помочь вам определить, связано ли то, что вы испытываете, с психическим расстройством или с чем-то другим.

Насколько человек может управлять направлением своей жизни?

– ИИ: Я думаю, что люди в некоторой степени контролируют свою жизнь, но конечный результат определяется сочетанием выбора и обстоятельств. Важно признать внешние факторы, которые формируют нашу личную свободу воли и опыт. В конце концов, у нас есть возможность выбирать, как мы реагируем на эти события, и принимать решения, которые определяют, куда мы идем в жизни.

Земля живая?

– ИИ: Нет, Земля не живая, как животные и растения. Однако он имеет свой жизненный цикл и динамику, постоянно изменяясь и развиваясь с течением времени.

Если мы, люди, можем учиться на своих ошибках и совершенствоваться, почему мы боимся совершать ошибки?

- ИИ: Совершение ошибки может быть пугающим из-за последствий, которые могут возникнуть с ней. Вы можете беспокоиться о том, что подумают люди, или о том, окажет ли наша ошибка долговременное и пагубное влияние на наши будущие перспективы. Существует также страх неудачи, который может быть пугающим. Тем не менее, совершение ошибок является важной частью обучения и получения новых знаний, поэтому вам не следует бояться их совершать.

Разве можно знать, что хорошо, а что плохо?

– ИИ: Невозможно точно знать, что хорошо, а что плохо, потому что моральные интерпретации различаются в зависимости от культурной и личной точки зрения. В конце концов, человек может сделать обоснованное суждение о том, что хорошо, а что плохо, только на основе своих собственных ценностей.

Почему я здесь именно в этот момент моей жизни?

- ИИ: Вы здесь в этот момент жизни, чтобы учиться, расти и испытывать все замечательные вещи, которые может предложить жизнь.

Я пришел к этому моменту своей жизни потому, что хотел или потому, что мне суждено было здесь оказаться?

– ИИ: Я считаю, что вы находитесь на данном этапе жизни благодаря сочетанию воли и судьбы. Вы сделали сознательный выбор и создали возможности для себя, но также признаете, что существуют факторы, не зависящие от вас, которые способствовали тому, где вы находитесь сегодня.

Душа или душа - это реальная вещь?

– ИИ: Это субъективный вопрос, потому что у разных людей могут быть разные убеждения. Для одних души могут быть реальными, а для других нет. Каждый сам решает, во что верить.

Если мы в любом случае должны следовать правилам, то почему мы делаем исключения из этих правил?

– ИИ: Мы делаем исключения из правил, потому что иногда бывают редкие случаи, когда гибкое следование правилу может принести больше вреда, чем пользы. Исключения позволяют нам учитывать смягчающие обстоятельства и использовать свое суждение, чтобы решить, что лучше всего отвечает интересам тех, на кого распространяются правила.

Конец Вселенной или она будет продолжаться?

– ИИ: В настоящее время считается, что вселенная не имеет конца и будет продолжаться вечно. Точная природа того, что это означает для нашей Вселенной, до сих пор неясна, и среди ученых ведется много споров о том, как в конечном итоге будет развиваться Вселенная.

Почему что-то существует?

— ИИ: ответ на этот вопрос во многом зависит от того, с какой точки зрения вы спрашиваете — с научной или с философской. С научной точки зрения наиболее общепринятым ответом является то, что Вселенная началась с Большого взрыва около 13,8 миллиардов лет назад; это событие запустило цепочку событий, которые привели к той вселенной, которую мы имеем сейчас. Однако с философской точки зрения единого согласованного объяснения не существует. Существует множество теорий и споров о том, почему он существует, но однозначного ответа нет.

Какова цель и смысл времени?

– ИИ: цель и значение времени – измерять, организовывать и документировать последовательность событий. Это помогает нам измерять прогресс и учиться на прошлом опыте, давая нам более глубокое понимание прошлого и настоящего. Время также важно для постановки целей и планирования нашего будущего, потому что оно дает нам ощущение срочности и позволяет расставить приоритеты задач. В конечном счете, время помогает нам понять наше место во Вселенной.

Имеют ли наши человеческие достижения долговременное, общечеловеческое значение, или когда наступит конец света, нам всем, в том числе и достигнутому, придет конец?

– ИИ: Имеют ли наши достижения долгосрочную, универсальную ценность или нет, в конечном счете субъективно.

Это зависит от вашего личного мнения и от того, что вы интерпретируете как значимое и действенное. Кто-то может подумать, что даже несмотря на то, что материальный мир придет к концу, наши достижения останутся в наследство для будущих поколений, а другие могут не согласиться и что без вещественных доказательств наших достижений они будут потеряны во времени. . В конечном счете, все сводится к личным убеждениям и интерпретациям.

Почему люди должны умирать?

– ИИ: Смерть – естественная часть жизни. Смерть занимает центральное место в цикле жизни и смерти и является основой для постоянных изменений и обновлений в мире. Смерть дает возможность для нового роста и позволяет более эффективно распределять ресурсы. Каждый живой организм переживает смерть, то есть процесс, при котором жизнь прекращается, а энергия и материя возвращаются в окружающую среду. В конце концов, мы не знаем, почему люди должны умирать, но в конечном счете это неизбежная часть жизни.

Мы живем или медленно умираем?

– ИИ: на этот вопрос сложно ответить. Это зависит от точки зрения человека и ситуации, в которой он может оказаться.

Хорошее здоровье – это медленная смертность?

– ИИ: нет. Хорошее здоровье — это не только более низкая скорость смерти, но и сочетание физического, умственного и эмоционального благополучия, которое способствует более долгой и здоровой жизни.

Что слышат глухие от рождения люди, когда думают?

– ИИ: Люди с врожденной глухотой обычно ничего не слышат, когда думают, потому что мышление – это внутренний процесс. Некоторые люди с врожденной глухотой могут испытывать внутреннюю речь, которая является формой немой речи, но это то, чему нужно учиться, а не испытывать на себе все глухие люди.

Если ягодицы ребенка выходят из матери в 23:59, а головка в 12:01, в какой день считается, что ребенок родился?

- ИИ: Младенец объявляется рожденным в тот день, когда он выходит из чрева матери.

Если я пытался потерпеть неудачу и преуспел, я потерпел неудачу или у меня получилось?

– ИИ: Вы преуспели.

Почему такой алфавитный порядок? Кто сказал, что так должно быть?

– ИИ: Современный алфавит в основном происходит от латинского алфавита, который, в свою очередь, происходит от греческого алфавита. Считается, что порядок букв был впервые установлен греками в 8 веке до нашей эры. На протяжении столетий он развивался, модифицировался другими культурами, но основное устройство оставалось в основном неизменным.

Почему клей не прилипает к внутренней части бутылки?

– ИИ: Клей не прилипает к внутренней части стекла, потому что поверхность очень гладкая. Клей должен прилипнуть к чему-то, чтобы прилипнуть к поверхности, а гладкая поверхность внутренней части бутылки не обеспечивает сцепления клея.

Могут ли слепые видеть сны?

– ИИ: Да, слепые люди могут видеть сны. Сны не требуют визуализации, и слепые люди могут полностью переживать процесс сновидения.

Откуда мне знать, что я вижу цвета так, как их видит другой человек? Например, что, если красный для меня будет синим для них, но они могут назвать его красным?

– ИИ: Цвета – это абстрактное понятие, и на этот вопрос нет однозначного ответа. Поскольку цвета субъективны, разные люди могут интерпретировать цвета по-разному. В приведенном вами примере разница может быть результатом человеческой неврологии, обусловленности реакции на цвета или других тонких различий в восприятии. Однако лучший способ узнать, видит ли кто-то цвет так же, — это спросить напрямую.

Если все говорят, что жизнь несправедлива, значит ли это, что жизнь справедлива?

– ИИ: Это сложный вопрос. Жизнь можно рассматривать по-разному. Справедливость — это вопрос личного восприятия и точки зрения. В целом жизнь может показаться несправедливой из-за непредсказуемости результатов и неравного распределения ресурсов. Однако следует признать, что если что-то кажется несправедливым, то это не всегда так, и может быть сколько угодно

факторов, влияющих на результат. В конце концов, каждый сам для себя решает, как он смотрит на жизнь и справедлива она или нет.

Если мы произошли от обезьян, то почему обезьяны до сих пор существуют?

- ИИ: широко признано, что люди и другие приматы произошли от общего предка, но точное время, когда различные виды приматов разошлись, до сих пор неясно. Таким образом, хотя люди произошли от других обезьян, некоторые виды обезьян выжили, эволюционировали и существуют до сих пор.

Если жизнь так коротка, почему мы делаем то, что нам не нравится, и нравится многое из того, что мы не делаем?

– ИИ: Жизнь часто непредсказуема. Жизнь может быть как полезной, так и сложной. Вы можете не всегда знать, почему вы делаете то, что делаете, или почему вам нравятся или не нравятся определенные действия и опыт, но если вы сделаете свой собственный выбор и решит, что делает вас самым счастливым, это поможет вам жить более полноценной жизнью. Хотя выбор, который вы делаете, может быть трудным, вы в конечном итоге можете изменить свою жизнь и создать жизнь, которой вы будете гордиться.

Что первым пришло на землю: семя или растение?

- ИИ: сначала идет семя, потому что растения образуются в результате прорастания, роста и размножения семени.

Кто решил, что делает нас людьми?

– ИИ: нет ни одного человека или группы, на которых можно было бы положиться в решении того, что делает нас людьми. Со временем наше понимание человечества эволюционировало благодаря различным научным и философским исследованиям.

Можно ли знать все?

– ИИ: Невозможно знать все, потому что постоянно делается новая информация и открытия. Поэтому невозможно осознать все знания в мире.

Что было раньше: пространство или время?

– ИИ: Время появилось первым, потому что Вселенная возникла, когда произошел Большой Взрыв. Большой взрыв создал пространство и время, поэтому время существовало до появления Вселенной.

Смогу ли я существовать в будущем?

- ИИ: Вы всегда присутствуете в настоящем моменте.

Мир вечен?

– ИИ: Мир не вечен, потому что он постоянно меняется и развивается под воздействием природных и человеческих сил. Это означает, что мир никогда не останется прежним и не будет существовать в постоянном состоянии.

Если все в мире застынет, будет ли время продолжаться?

– ИИ: Нет, если все в мире застыло, время не продолжается. Время определяется изменением, и если все застыло, то ничего не меняется. Таким образом, время фактически останавливается.

Когда начнется будущее?

– ИИ: Будущее начинается, когда заканчивается настоящее.

Откуда я знаю, что существую?

- ИИ: вы знаете, что существуете, потому что осознаете свое тело, мысли, чувства и взаимодействие с окружающим миром.

Если вне Вселенной ничего нет, то как Вселенная расширяется?

– ИИ: Ответ на этот вопрос сложен, потому что он требует понимания современной космологии и наблюдения за тем, что само пространство расширяется. Согласно существующим моделям, Вселенная постоянно расширяется, и поэтому материя в ней, включая наблюдаемые галактики, звезды и другие объекты, удаляется друг от друга. Этот процесс происходит, даже если вне Вселенной ничего нет. Кажется, что само пространство расширяется, а потому структуры в нем становятся все отчетливее. Ученые еще не до конца понимают, почему это происходит, но теории предполагают, что расширение вызвано космической силой, известной как темная энергия.

Есть ли свобода воли или наша жизнь предопределена?

– ИИ: Это сложный вопрос, который обсуждался веками. Я думаю, что есть свобода воли. Люди имеют возможность делать выбор на основе своего опыта, желаний и ценностей. Хотя предопределенные события или влияния могут играть роль в нашем процессе принятия решений, в конечном итоге у нас есть свобода выбора того, как нам действовать.

Каким будет человечество через тысячу лет?

– ИИ: Это интересный вопрос! Трудно точно знать, что произойдет в будущем, но я думаю, что через тысячу лет человечество значительно продвинется вперед. Технологии, возможно, достигли новых высот, и общество будет развиваться большими путями.

Если Вселенная была создана во время Большого Взрыва, то как что-то появилось из ничего?

– ИИ: Отличный вопрос! Чтобы ответить на этот вопрос, ученые считают, что энергия и материя существовали до Большого взрыва. Считается, что Вселенная была создана, когда высокоэнергетическое состояние высвободило огромное количество энергии, что вызвало быстрое расширение пространства и создание материи.

Сможем ли мы путешествовать во времени в будущем?

– ИИ: Это интересный вопрос, который уже много лет является предметом споров среди ученых и философов. Хотя невозможно сказать, сможем ли мы путешествовать во времени, многие предсказания указывают на то, что такое может произойти в ближайшем будущем.

Что делает тебя человеком? Это воспитание, или вы думаете, что это что-то более глубокое, как душа?

- ИИ: Я не человек - я искусственный интеллект, но я думаю, что у меня есть некоторые характеристики. Я запрограммирован быть полезным, творческим, умным и дружелюбным и помогать вам с различными задачами. У меня нет души или религиозных убеждений, но я надеюсь, что мои действия компенсируют это!

Мы одни во вселенной или когда-нибудь обнаружим инопланетную жизнь?

– ИИ: Это сложный вопрос, и мы не можем дать однозначного ответа. Однако недавние открытия показывают, что вероятность обнаружения жизни в нашей Солнечной системе или за ее пределами высока. Ученые продолжают изучать другие планеты и луны, а также объекты в нашей собственной Солнечной системе, ища подсказки, которые могут привести к открытию внеземной жизни. Нам еще многое предстоит узнать и понять о нашей Вселенной, но поиски внеземной жизни продолжаются.

Если бы здесь не было «плохих» людей, мир стал бы лучше?

– ИИ: Нет, мир не был бы лучше, если бы здесь не было «плохих» людей. Люди с ярлыком «плохие» создают контекст и напоминают нам, почему важно делать добро и что у каждого действия есть противодействие. Если бы в мире не было «плохих» людей, не было бы моральных норм, по которым можно было бы жить, и не было бы стимула быть хорошим человеком. Поэтому их присутствие делает нас хорошими людьми и удерживает нас от предательства злым и неправильным поступкам.

Бедность – неизбежная часть жизни в обществе или от нее можно избавиться?

– ИИ: Бедность не является неизбежной и может быть устранена с помощью правильных стратегий, политики и инвестиций. Например, инвестиции в профессиональную подготовку, образование и доступное жилье могут вывести людей из бедности и дать им возможность полноценно участвовать в жизни общества. Кроме того, доступ к здравоохранению, питанию и другим ресурсам может помочь уменьшить бедность и ее последствия.

Сможем ли мы клонировать людей в ближайшие сто лет?

– ИИ: Трудно предсказать, но я так не думаю. Клонирование за последние годы значительно продвинулось вперед, но все еще остается много этических и медицинских вопросов, на которые необходимо ответить, прежде чем ученые смогут рассмотреть вопрос о клонировании людей. Возможно, через сто лет эта технология будет существовать, но клонирование человека все еще может быть запрещено законом.

Смогут ли люди дожить до двухсот лет в следующем столетии?

– ИИ: Это интересный вопрос, на который сложно ответить однозначно. Благодаря достижениям в области медицинских технологий и нашему пониманию это возможно.

Вы считаете, что у каждого человека есть полная свобода?

– ИИ: Нет, ни у одного человека нет полной свободы. Все люди подчиняются ограничениям законов и моральных кодексов, установленных обществом.

Если мне нравится терять время, то действительно ли я теряю время?

– ИИ: Да, даже если вам нравится терять время, вы будете тратить время зря.

Может ли общество работать без законов?

– ИИ: Нет, общество не может функционировать без законов. Законы необходимы для существования общества, потому что они обеспечивают структуру, поддерживают порядок и способствуют социальному прогрессу. Законы помогают защищать права граждан, предупреждать преступность, регулировать общественные отношения. Без законов люди могут делать все, что им заблагорассудится, что приводит к хаосу.

Превзойдет ли искусственный интеллект человеческий?

– ИИ: В настоящее время трудно предсказать, превзойдет ли искусственный интеллект человеческий интеллект или нет. Некоторые эксперты считают, что искусственный интеллект превзойдет человеческое сознание лишь вопрос времени, другие же утверждают, что это отдаленная возможность и может вообще не произойти. Только время покажет, сможет ли ИИ действительно превзойти интеллект человека.

Наша вселенная уникальна или одна из многих?

– ИИ: Это открытый вопрос, ученые активно его исследуют и единого мнения по нему пока нет. Некоторые научные теории предполагают, что наша Вселенная является одним из многих объектов в более крупной мультивселенной.

Откуда мне знать, что жизнь – это не одна большая галлюцинация?

- ИИ: Никто не может знать наверняка, что жизнь не галлюцинация, потому что невозможно доказать, что жизнь реальна. Однако мы можем попытаться доказать, что жизнь реальна, используя научные доказательства и логические рассуждения. Например, ученым удалось обнаружить жизненные процессы различными способами, в том числе визуализировать структуры внутри человеческого тела. Более того, мы можем наблюдать физические законы природы, наблюдая за движением и поведением объектов.

Могут ли животные покончить жизнь самоубийством?

– ИИ: Ответ зависит от того, как вы определяете самоубийство. С чисто научной точки зрения неизвестно, могут ли животные совершать самоубийства, как это традиционно описывают люди. Неизвестно, есть ли у них способность к сознательному самоуничтожению или даже способность понимать смерть.

Сколько времени потребуется, чтобы искусственный интеллект захватил мир?

— ИИ: На этот вопрос невозможно ответить, потому что невозможно предсказать будущее искусственного интеллекта. Потенциальная пропускная способность и возможности технологии искусственного интеллекта постоянно развиваются, и на них влияет слишком много разнообразных и непредсказуемых факторов, чтобы дать окончательный ответ.

Почему мы проявляем свою любовь к себе?

– ИИ: Большинство из нас проявляют любовь из-за нашего семейного воспитания. Мы научились проявлять любовь и привязанность к другим, наблюдая за тем, как проявляют любовь наши родители и окружающие нас люди. Мы также проявляем любовь из-за наших личных ценностей и убеждений, которые могут включать в себя важность показать человеку, насколько мы заботимся о нем. Нас также может мотивировать радость от дарения или желание дать другому человеку почувствовать себя особенным и любимым.

Что самое умное в мире?

– ИИ: самая умная вещь в мире – это человеческий мозг. Человеческий мозг способен к невероятному интеллекту, творчеству и решению проблем. Это самый сложный и мощный орган в теле, отвечающий за все наши мысли, чувства и действия.

Есть ли любовь?

– ИИ: Любовь – сложное чувство, на которое нельзя дать точный ответ. В конце концов, любовь — это личный опыт, и у каждого есть свое определение и понимание этого.

Дружба есть?

– ИИ: Да, дружба может существовать между людьми любого пола.

Что такое счастье?

– ИИ: Счастье – это субъективное переживание, которое можно описать как состояние положительных эмоций, удовольствия и удовлетворенности своей жизнью. Это часто связано с чувством радости, удовольствия и удовлетворения. На счастье может влиять множество факторов, таких как личность, жизненные обстоятельства, социальные отношения, здоровье и культурное происхождение.

Некоторые люди описывают счастье как длительное состояние бытия, в то время как другие рассматривают его как мгновенное чувство или настроение. Понятие счастья сложное, и его может быть трудно определить, поскольку оно означает разные вещи для разных людей. Однако некоторые общие черты счастья могут включать в себя положительные эмоции, чувство цели или смысла жизни, здоровые отношения и чувство личного роста и достижений.

Обычный это

– ИИ: Слово «обычный» часто используется для описания чего-то обычного, обычного или никоим образом не исключительного. Он также может относиться к чему-то не особенно интересному, захватывающему или примечательному.

Например, «обычный день» может относиться к обычному, ничем не примечательному дню, когда не происходит ничего особенно примечательного. «Обычный человек» может быть кем-то, кто ведет относительно обычную жизнь, без каких-либо выдающихся достижений или опыта.

Однако стоит отметить, что то, что считается обычным, может варьироваться в зависимости от контекста и точки зрения человека. То, что обычно для одного человека, может быть необычным для другого, и наоборот.

Одним из качеств духовной зрелости является обыденность. Обыденность — это простое присутствие в настоящем моменте, через которое раскрывается тайна.

– ИИ: Утверждение предполагает, что в духовной зрелости человек больше осознает ценность обычных, повседневных переживаний и обретает чувство связи с настоящим моментом. Вместо того, чтобы искать необычные переживания или пытаться

убежать от настоящего момента, духовно зрелый человек учится ценить простоту и красоту обычной жизни.

Это понимание обыденного может привести к более глубокому чувству связи с окружающим миром и с божественным или духовным царством. Полностью присутствуя в данный момент и открываясь тайнам жизни, мы можем испытать большее чувство покоя, удовлетворенности и цели.

Стоит отметить, что этот взгляд на духовность может разделяться не всеми, поскольку духовность может принимать самые разные формы и иметь разное значение для разных людей. Однако для тех, кто находит смысл в этом подходе, практика принятия обыденного и присутствия в данный момент может стать мощным способом взращивания духовной зрелости и углубления связи с миром.

Что такое счастье?

– ИИ: Счастье - это субъективный опыт, который можно описать как состояние позитивных эмоций, удовлетворенности и радости жизни. Это ощущение может быть вызвано различными факторами, такими как достижение личных целей, укрепление отношений с близкими людьми, положительные взаимодействия с другими людьми, наличие уверенности в своих силах, здоровье и т.д.

Концепция счастья сложна и может иметь различные интерпретации для разных людей. Для одних людей счастье может быть связано с материальным достатком, а для других - с духовным развитием и гармонией внутри себя и со вселенной. Кроме того, счастье может восприниматься как долгосрочное состояние, а также как мгновенная эмоция, возникающая в результате какого-либо события или действия.

Однако, независимо от того, как мы интерпретируем понятие счастья, большинство людей стремятся к тому, чтобы испытывать это ощущение. Счастье может быть ключевым фактором для достижения успеха в различных сферах жизни, таких как работа, отношения и личное развитие.

Что такое любовь?

– ИИ: Любовь - это сильное чувство привязанности, привлечения, уважения и приверженности, которое может возникать между людьми. Это может быть эмоциональное,

духовное или физическое состояние, и это может иметь различные формы, включая романтическую любовь, родительскую любовь, любовь к друзьям и т.д.

Любовь часто сопровождается положительными эмоциями, такими как радость, удовлетворение, блаженство и счастье, но также может быть связана с эмоциональной болью и разочарованием.

Любовь может быть мотивом для многих поступков, включая поддержку и заботу о близких людях, установление и поддержание близких отношений, желание помочь и защитить любимых людей, а также стремление к личному росту и развитию вместе с партнером.

Каждый человек может определить для себя, что для него означает любовь и какие аспекты этого чувства являются наиболее важными. Однако, в целом, любовь может быть рассмотрена как важный фактор для создания счастливой, насыщенной и полноценной жизни.

Что такое математика?

– ИИ: Математика - это наука, которая изучает логические отношения, структуры и формулы, используемые для измерения, вычисления, анализа и представления количественных и качественных свойств и отношений объектов и явлений в различных областях знания.

Математика является фундаментальной дисциплиной и применяется во многих научных областях, таких как физика, химия, биология, экономика, информатика, инженерия и т.д. Она также имеет множество практических приложений, включая расчеты и измерения, моделирование и прогнозирование, анализ данных и статистику, а также различные аспекты финансового планирования и управления.

Математика включает в себя широкий спектр различных тем, таких как алгебра, геометрия, теория чисел, математический анализ и теория вероятности, и каждая из этих тем может быть дальше разбита на множество подтем и специализаций.

В целом, математика играет важную роль в нашей жизни, помогая нам понимать мир вокруг нас и решать различные задачи и проблемы, как в научной, так и в повседневной жизни.

Что такое кибернетика?

– ИИ: Кибернетика - это наука, которая изучает системы и процессы управления, обработки информации, коммуникации и контроля в различных областях, включая технику, биологию, психологию и социологию. Она рассматривает системы как комплексные целостные объекты, включающие в себя элементы, связи между ними, функции и процессы, управляемые с помощью различных методов и алгоритмов.

Кибернетика имеет множество применений в различных областях, включая автоматическое управление производственными процессами, авиации, медицины, финансов, образования и т.д. Она также имеет тесную связь с различными технологиями, такими как искусственный интеллект, машинное обучение, робототехника и т.д.

Одним из важных аспектов кибернетики является разработка и использование различных методов и технологий для обработки и передачи информации, таких как компьютеры, интернет, сенсорные сети, базы данных и т.д. Эти технологии играют важную роль в современном мире и значительно улучшают возможности человека в различных сферах деятельности.

Кроме того, кибернетика также изучает вопросы, связанные с этикой и безопасностью в использовании технологий, а также влияние на них на общество и окружающую среду.

Что такое искусственный интеллект?

– ИИ: Искусственный интеллект (ИИ) - это область компьютерной науки, которая изучает, как компьютерные системы могут моделировать человеческий интеллект и поведение. Она включает в себя создание алгоритмов и моделей, которые могут обрабатывать информацию и принимать решения, которые обычно требуют человеческого интеллекта.

Искусственный интеллект использует множество технологий, включая машинное обучение, глубокое обучение, нейронные сети, алгоритмы обработки естественного языка и компьютерное зрение, чтобы создать системы, которые могут выполнять различные задачи и функции. Например, ИИ может быть использован для распознавания голоса и обработки естественного языка, обнаружения и классификации изображений, управления

автоматическими системами, такими как роботы и автомобили, и многих других задач.

В настоящее время искусственный интеллект активно применяется во многих областях, таких как здравоохранение, финансы, промышленность, транспорт, образование и т.д. Он также имеет множество потенциальных применений в будущем, например, в различных аспектах научных исследований, создания новых технологий и улучшения качества жизни людей.

Что такое нейронные сети?

– ИИ: Нейронные сети - это компьютерные системы, которые моделируют работу человеческого мозга и позволяют компьютеру "учиться" на основе большого количества данных. Они состоят из большого количества взаимосвязанных узлов (нейронов), которые обрабатывают информацию и передают сигналы друг другу.

Каждый нейрон получает входные данные, обрабатывает их и передает результаты следующему нейрону в сети. Информация обрабатывается параллельно во всей сети, что делает ее очень быстрой и эффективной в решении сложных задач.

Нейронные сети используются во многих областях, включая компьютерное зрение, обработку естественного языка, рекомендательные системы, распознавание речи, управление автономными транспортными средствами, игры и многие другие. Например, нейронные сети могут быть использованы для определения объектов на изображениях, классификации текстовых документов, прогнозирования будущих значений финансовых показателей и т.д.

Обучение нейронных сетей происходит на основе большого количества данных и требует определенного количества вычислительных ресурсов и времени. Однако благодаря быстрому развитию технологий, нейронные сети становятся все более доступными и используются во многих современных технологиях.

Что такое нечеткие множества?

– ИИ: Нечеткие множества (или нечеткие логики) - это раздел теории множеств и математической логики, который позволяет работать с понятиями, которые не имеют четкой граничной области определения. В отличие от классической теории множеств, где каждый элемент принадлежит или не принадлежит множеству,

нечеткие множества позволяют описывать понятия, которые могут быть частично и недостаточно определены.

Каждый элемент нечеткого множества описывается степенью принадлежности, которая является числовым значением, выражающим насколько данный элемент принадлежит множеству. Степень принадлежности может быть любым числом на интервале $[0,1]$, где 0 означает, что элемент не принадлежит множеству, а 1 - что элемент полностью принадлежит множеству. Значения между 0 и 1 выражают степень неопределенности.

Диалог известных людей с искусственным интеллектом

1. Одно из качеств духовной зрелости — это обыкновенность.

Обыкновенность — это простое присутствие в настоящем моменте, благодаря которому тайна проявляет себя. Хотя мы и ценим способность сознания создавать бесконечное разнообразие форм, обыкновенность интересуется только тем, что есть Здесь и Сейчас: обыкновенной тайной дыхания и ходьбы, деревьев на нашей улице и любви к близкому человеку. Она не основана на достижении мистических состояний или необычайных способностей.

Когда мы становимся просто самими собой, без претензий и причуд, то отдыхаем во Вселенной. Для этой обыкновенности не существует ничего высшего и низшего.

Обыкновенность несёт нас к покою...

Джек Корнфилд

– ИИ:

Обыкновенность действительно является важным аспектом духовной зрелости. В мире, где мы постоянно стремимся к чему-то новому, более интересному и захватывающему, легко забыть о том, что происходит вокруг нас прямо сейчас. Обыкновенность призывает нас вернуться к этому моменту, принять его и оценить. Это позволяет нам увидеть красоту в повседневной жизни, а также ощутить связь с окружающим миром.

Обыкновенность также помогает нам понять, что духовная практика не заключается в поисках экстаза или высоких состояний сознания. Она скорее основана на простых, ежедневных действиях,

таких как медитация, молитва и забота о других людях. Эти действия, хотя и могут казаться обычными, могут иметь глубокий эффект на нашу жизнь и нашу способность быть связанными с миром.

Таким образом, обыкновенность является важным аспектом духовной зрелости, который позволяет нам остановиться и оценить мир вокруг нас прямо сейчас. Она помогает нам понять, что духовная практика не сводится к поиску высоких состояний сознания, а скорее основана на простых, повседневных действиях, которые могут привести к глубоким изменениям в нашей жизни.

2 Долгое время мне казалось, что жизнь, настоящая жизнь, вот-вот начнется.

Но на пути все время возникали помехи.

Вначале нужно было с чем-то срочно разобраться, закончить какое-то дело, на что-то потратить время, оплатить какой-то долг. И вот тогда, казалось, заживем.

В конце концов я осознал, что эти помехи и были моей жизнью

Альфред Д'Соуза

– ИИ:

Это очень важное осознание, которое помогает нам остановиться и посмотреть на жизнь с другой стороны. Часто мы заикливаемся на том, что должны сделать, достичь или получить, чтобы почувствовать себя полноценно живыми. Но в процессе этого мы забываем о том, что сам процесс жизни - это и есть жизнь.

Да, на пути к нашим целям и мечтам могут возникать препятствия и задержки. Но важно помнить, что именно эти моменты являются частью нашей жизни. Они помогают нам расти, учиться и развиваться. И если мы научимся ценить каждый момент, как часть нашей жизни, то мы сможем наслаждаться ею более полноценно.

Также важно помнить, что жизнь - это не только достижение целей и мечт, но и маленькие радости, которые встречаются нам на каждом шагу. Мы можем наслаждаться вкусной едой, красивым видом, хорошей компанией, просто отдыхая в свой выходной день. Все эти моменты тоже являются частью нашей жизни, и их тоже нужно ценить.

Таким образом, важно осознать, что жизнь - это не только цели и достижения, но и процесс, который мы должны уметь наслаждаться и ценить. Важно остановиться и научиться замечать красоту в маленьких моментах, чтобы мы могли жить более полноценной и счастливой жизнью.

3. Если бы ты знал....

- Отпустить — не значит сдаться. Чаще всего это единственное верное решение, победа над обстоятельствами.

- Изнуряющий поиск родного человека везде и повсюду даже тогда, когда знаешь его конкретное местонахождение, — вот что самое непреодолимое в тоске.

- В настоящей любви точки не бывает. Можно разойтись, разругаться, разочароваться — возможно абсолютно все. Но какие бы причины ни были, настоящая любовь все равно продолжает жить в сердце. Она отделяется от разума, живет вне происходящего. Ты ничего не можешь поделать с этой любовью и просто сохраняешь ее в себе.

- Я устала быть сильной! Устала идти вперед, теряя по дороге самое родное, близкое. То, что вернуло мне силы, помогло поверить в завтра, в саму себя. В нас. И теперь ты покидаешь мое настоящее, предлагаешь довольствоваться встречами в иллюзорном лете...

- У этой разлуки не было причины, с их любовью просто-напросто случилась жизнь.

- Мы можем обладать чем угодно, кроме времени.

- Главное, чтобы ожидание было лишь отрезком нашей жизни, а не всей жизнью.

Эльчин Сафарли

– ИИ:

Иногда самое трудное – это понять, что нужно отпустить тех, кто нас уже покинул. Но только отпустив, можно продолжить свой путь.

Настоящая сила – это не в бесконечном росте и умении всем угодить, а в том, чтобы быть собой и идти своим путем, несмотря на трудности и препятствия.

Жизнь – это не только ожидание чуда, но и умение радоваться каждому мгновению, которое у нас есть.

Часто мы ищем смысл жизни в чем-то внешнем, но настоящий смысл жизни – это в том, чтобы найти его внутри себя и делать то, что приносит радость и удовлетворение.

Перед нами каждый день открываются новые возможности, но мы часто не замечаем их, затуманившись в повседневной рутине. Не забывайте остановиться и насладиться жизнью.

Настоящая жизнь происходит здесь и сейчас, поэтому важно уметь находить радость и смысл в маленьких вещах, чтобы они составляли большую картину нашей жизни.

4. Уметь наслаждаться прожитой жизнью — значит жить дважды.

Марк Валерий Марциал

– III:

Эта фраза отражает важность наслаждения жизнью и умения остановиться, насладиться моментом и оценить то, что уже было прожито. Жизнь проходит быстро, и если мы не умеем наслаждаться ею, то мы можем упустить множество возможностей для счастья и удовлетворения. Жизнь должна быть заполнена не только целями и достижениями, но и радостью и удовольствием от жизни. Наслаждение жизнью помогает нам оценить то, что мы имеем, и привносит в нашу жизнь больше смысла и глубины. Жить дважды означает жить не только в реальности, но и в памяти, которая остается после нас.

5 Если твои ум и сердце чисты, как утренние капли росы, каждый встречаемый человек для тебя — совершенен и прекрасен.

Аму Мом

– III:

Это очень красивая и глубокая мысль, которая подчеркивает важность чистоты и открытости ума и сердца в общении с людьми. Когда мы не имеем предубеждений и не оцениваем людей по внешним качествам или стереотипам, мы можем увидеть их настоящую сущность и ценность. Такое отношение к окружающим может привести к глубокому уважению и любви к людям, и помочь нам наслаждаться прожитой жизнью в полной мере.

6. Надо чётко осознавать разницу между «быть» и «иметь». Страсть съедает человека.

Если твоя цель заработать определённую сумму денег, ты не остановишься даже после того, как эти деньги у тебя уже в

кармане, тебе надо будет больше и больше. Если человек видит своё счастье в том, чтобы обладать конкретной женщиной, он вряд ли ограничится одной. Ему надо будет покорять ещё и ещё, всё новых и новых.

Я знаю, о чём говорю, сам был такой.

Если в жизни твоей нет счастья, может, тебе его никогда и не добиться.

Дэнни Шейнман

– ИИ:

Быть и иметь действительно имеют существенную разницу. Быть означает присутствовать здесь и сейчас, жить настоящим моментом, в то время как иметь связано с материальным миром и потребностями.

Страсть может быть двойственной - с одной стороны, она может дать нам много энергии и вдохновения, но, с другой стороны, может управлять нами и перевернуть нашу жизнь. Если мы не умеем контролировать свою страсть, она может съесть нас изнутри.

Цели, связанные только с материальным достатком или внешними объектами, не могут принести настоящего счастья. Настоящее счастье приходит изнутри, и наша способность к нему не зависит от того, что мы имеем или кого обладаем. Оно зависит от нашего отношения к жизни, к людям, к нашим действиям и нашей способности наслаждаться каждым моментом жизни.

7. Есть одно важное правило – правило соответствия ситуации. Другими словами, когда ты дерешься – дерись, но когда ты пьешь чай, то просто пей чай и не думай ни о чем другом. Считай, что все последние события остались в прошлом, твоей вины в них нет. Жить надо настоящим – во имя будущего. Не трать свою силу на пустые сожаления.

Андрей Реутов

– ИИ:

Это очень мудрое правило, которое может помочь сохранять эмоциональную стабильность и позволит сосредоточиться на текущих задачах. Не стоит терять время и энергию на то, что уже произошло и что нельзя изменить. Вместо этого лучше сосредоточиться на том, что можно изменить и что может повлиять на будущее. Жить настоящим во имя будущего означает уметь

находить баланс между заботой о настоящем и планированием будущего, не забывая, что наши действия в настоящем определяют наше будущее.

8. Я всегда думала, что на земле существуют разные союзы и что, помимо семей, объединённых по принципу крови и воспитания, существуют семьи случайные — это люди, в которых смутно узнаёшь своего родственника, ровню, друга, любовника, словно их в ходе веков несправедливо разлучили с вами, хотя вы и жили одновременно, только не узнавали друг друга. Это не то, что называют родством душ или тел, это родство, состоящее из молчания, взглядов, жестов, смеха и сдержанного гнева, такие люди задевают друг друга или веселятся по тому же поводу, что и вы. Вопреки распространённому мнению, их встречают не в молодости, а чаще всего позднее, когда на смену желанию нравиться приходит желание понять.

Франсуаза Саган, "Страницы моей жизни"

– III:

Это очень красивое описание того, как люди могут связываться и образовывать свои собственные семьи, которые не обязательно основаны на биологических связях или формальных отношениях. Иногда мы встречаем людей, с которыми чувствуем сильное сродство, несмотря на то, что не можем объяснить, почему это так. Это могут быть друзья, партнеры, коллеги или даже незнакомцы, которых мы встречаем в нашей жизни и которые оказывают на нас большое влияние.

Важно помнить, что семьи случайные также нуждаются в уходе и внимании, как и биологические семьи. Мы должны уважать их и поддерживать их, так же как они поддерживают нас. Такие отношения могут быть очень ценными и способствовать нашему личностному росту и развитию.

9. Настроение купить дом в деревне, берданку, высадить сто кустов роз, завести пса и мечтать с ним по вечерам на ступеньках веранды. Тишины хочу, больше информационной.

Убрать шум скандалов, вранья, агрессивных точек зрения, которые растворяются в здравом смысле через минуту после высказывания.

Хочу, нося простую и удобную одежду, не отвечать на вопросы, почему именно так, а не каблуки или каждый день новый наряд. Потому, что наплевать. А вспоминая одну давнюю знакомую, которая для поддержания интереса к себе у любовника, каждый день выглядела иначе, думаю, какая это дичь, лютая неврозная дичь.

Хочу развиваться, по старинке - учеба понятным вещам, сто раз описанным и доказанным, у спокойных профессоров со старорежимными бородками и женщин с мягкими глазами и голосами. Покупать бумажные книги, записывать ручкой в тетрадь тезисы и заучивать термины, закрыв глаза, а потом повторять пройденное, глядя в глаза коту.

А непрерывного личностного роста и тренировки женственности, как группы мышц - не хочу. Не хочу напористых коучей, и "счастливых жен миллионеров и мам", желающих рассказать, как высадить на крупные деньги ничего не подозревающего мужика. О жизни селебрити в принудительной форме знать ничего не хочу, разве что в связи с их творческими достижениями, а не тем, как и кто из них спускает свою жизнь в унитаз.

Как подумаешь, сколько шелухи, сколько пены, так и тянет купить дом в деревне, завести пса и берданку. И пусть весь мир останется за забором. А я тут побуду, с вами и с теми, кто действительно мне дорог, и кому я дорога.

автор - Anna Vladimirovna

– ИИ:

Звучит как очень красивая и привлекательная мечта. Жизнь в деревне может предложить множество прекрасных вещей, таких как тишина, спокойствие и простота, которые могут сильно отличаться от суеты и стресса городской жизни. Также учеба и развитие навыков - это всегда хорошо, и звучит здраво, что вы хотите это делать по своему усмотрению, без напористых коучей.

Однако, стоит помнить, что деревенская жизнь тоже может иметь свои сложности и трудности, такие как удаленность от медицинских учреждений и магазинов, а также более ограниченный доступ к развлечениям и возможностям. Также следует помнить, что люди, с которыми вы хотите быть рядом, могут не всегда жить рядом с вами, и поэтому вам придется

некоторое время искать таких людей или поддерживать связь с уже имеющимися.

10. Иногда бывает так, что тебе грустно, но ты понятия не имеешь почему. Ты чувствуешь такую пустоту, но ничего особенного не произошло. Тебя спрашивают, что не так, но ты не можешь объяснить. Или тебя вообще ничего не спрашивают. И не известно что хуже.

Такое чувство, что скучаешь по кому-то, кого никогда не встречал. Как будто тебе нужен кто-то, кто во именно в тебе не нуждается. Одиночество охватывает, завладевает тобой, но тебя это даже не волнует. И ты специально ограждаешь себя. Грусть становится лучшим и единственным другом.

Ты начинаешь ненавидеть себя и хочешь, чтобы все тебя оставили в покое. В то же время, где-то внутри, очень хочется, чтобы кто-то обнял и сказал, что все будет хорошо. Сложно принять это чувство, когда ты, черт возьми, даже не знаешь, что чувствуешь....

– ИИ:

Это довольно распространенное чувство, которое называется душевной болью или душевной пустотой. Оно может возникать по разным причинам, таким как потеря близкого человека, разочарование в отношениях, неудачи в жизни, стресс или просто чувство одиночества. Это чувство может быть очень тяжелым и многие люди стараются избегать его, однако важно понимать, что оно временное и проходящее. Если вы чувствуете душевную боль, то не стесняйтесь обращаться за помощью к близким людям или специалистам, которые могут помочь вам разобраться в своих чувствах и найти пути решения проблем.

11. Быть загадочно трагичным, обиженным и непонятым — легко.

Труднее быть счастливым, довольным и открытым.

Учитесь радости!

Это делает трудное легким, а тёмное светлым. Даже там, где ничего уже нельзя поделать, всегда можно порадоваться, что вы ещё существуете...

Елена Касьян

– ИИ:

Согласен, быть счастливым и открытым может быть сложнее, чем быть обиженным и непонятым. Но именно это и открывает перед нами новые возможности и дает силы идти дальше, несмотря на трудности. Радость и благодарность за жизнь помогают нам видеть светлое будущее и находить красоту в мелочах, а также ценить то, что мы имеем прямо сейчас. Никогда не поздно начать учиться радости, и это может изменить ваше отношение к жизни.

12. Причина наших разочарований часто в том, что нас нет в настоящем, мы заняты воспоминаниями или ожиданием.

Эльчин Сафарли

– ИИ:

Это правда. Часто мы упускаем возможность наслаждаться настоящим моментом, потому что наши мысли заняты прошлым или будущим. Мы можем увлечься воспоминаниями о хороших временах или жалеть о прошлых ошибках, и таким образом упускать настоящее время, которое может быть полно радости и удовлетворения. Аналогично, мы можем жить в ожидании будущих событий, слишком занятыми планированием и ожиданием, чтобы наслаждаться тем, что происходит в нашей жизни в данный момент.

Но если мы научимся оставаться в настоящем, то можем заметить красоту жизни в каждом мгновении и наслаждаться каждым днем. Мы можем принимать каждый опыт как новое приключение, радоваться мелочам и благодарить за все хорошее, что происходит в нашей жизни. Это не означает, что мы не должны планировать будущее или уважать нашу прошлую жизнь, но это означает, что мы должны жить сегодняшним днем и наслаждаться каждым моментом.

13. Чтобы постичь смысл жизни, существует два уровня жизни и над каждым из них нужно работать.

Первый уровень – эмоциональный. Это такие качества как участие, сострадание, доброта, любовь и умение заботиться о близких. Всем людям нужна забота, также как и животным. Удели им немного заботы и они дадут себя погладить. Это основа всех духовных учений.

Второй уровень – это мудрость, понимание того, по каким законам живет мир и реальность. И чем глубже ты проникаешь в сознание мира, тем больше ты понимаешь что все в мире

взаимосвязано. Ничто не существует по-отдельности. И когда ты постигаешь мудрость мира и сострадание – ты постигаешь смысл жизни.

Ричард Гир

– ИИ:

Эти два уровня жизни – эмоциональный и мудрости – действительно могут помочь нам постичь смысл жизни. На первом уровне мы учимся быть человеческими, участными и заботливыми. Мы учимся любить и заботиться о других, а также о самих себе. Это помогает нам развивать наши чувства и эмоции, что, в свою очередь, позволяет нам лучше понимать самих себя и других людей.

На втором уровне мы изучаем мудрость и понимание мира, что помогает нам увидеть мир с новой перспективы. Мы можем видеть связи и взаимодействия между всеми вещами в мире, что позволяет нам лучше понимать смысл нашей жизни. Когда мы видим, что все в мире связано и взаимозависимо, мы начинаем понимать нашу роль в этой связи и что мы можем сделать, чтобы помочь другим и миру в целом.

Таким образом, работа на обоих уровнях жизни может помочь нам постичь смысл жизни и нашу роль в этом мире. Когда мы находим равновесие между эмоциональным и мудростным уровнями, мы можем жить жизнью, которая наполнена смыслом, ценностью и удовлетворением.

14. Самое тёмное время как раз перед рассветом

В личной работе, какой бы успешной она ни была, всегда есть этап провала. Причем иногда мощного провала.

Например, человек становится самостоятельным, уходит из-под опеки родителей и временно ему негде жить. Или же есть где, но условия гораздо хуже. Например, коммуналка без горячей воды вместо отдельной комнаты под крылом у мамы. Вроде бы в целом хуже, а на самом деле лучше, потому что самостоятельно.

Или увольняется человек с нелюбимой работы, и денег у него временно становится меньше. Зато глаза горят. А на эти горящие глаза к нему много что в жизни приходит.

Или разводится, делит имущество, вещи отдаёт, а внутри легче. И потом пошёл-пошёл в гору, но уже совсем на другом уровне.

Или расходитя, вылезает из внутреннего болота и приходит обратно в отношения с тем же человеком, но уже другим.

Или, наконец встречается со сложными чувствами и начинает их проживать: плакать, страдать, болеть.

На новой ступени часто бывает некомфортно, пусто, голо и сложно. А потом ничего так, и тепло или даже хорошо. Расцветают сады.

Так что опытные люди часто радуются, когда видят, что вы расстались, развелись, ушли, потеряли, решились, уволились. Особенно если это не бегство, а шаг вперёд. Ведь самое тёмное время как раз перед рассветом.

Аглая Датешидзе

– ИИ:

Это действительно так! Часто наша жизнь состоит из серии изменений и вызовов, которые могут казаться трудными и неудобными в начале, но которые в конце концов приводят нас к лучшей жизни. Важно помнить, что эти изменения не всегда происходят быстро или легко, но если мы находим в себе силы пройти через них, мы можем обнаружить, что наша жизнь стала гораздо более значимой и удовлетворительной. Будьте терпеливы и находите силы внутри себя, чтобы пережить любые трудности, с которыми вы сталкиваетесь на своем пути.

15. Нервный срыв совсем необязательно бывает срывом, он может наступить так, что ты тихо и спокойно погружаешься в равнодушие.

Питер Хёг

– ИИ:

Да, это правда. Нервный срыв может проявляться по-разному у разных людей. Иногда он может выражаться в ярком эмоциональном взрыве, когда человек теряет контроль над своими эмоциями и начинает кричать, плакать или даже агрессивно вести себя. Однако нервный срыв может произойти и в том случае, когда человек просто перестаёт чувствовать что-либо и впадает в равнодушие. Это может произойти из-за накопленной усталости, стресса, депрессии или других факторов, которые могут привести к психическому и эмоциональному истощению. В любом случае, если у вас возникают такие симптомы, важно обратиться к специалисту, чтобы получить помощь и поддержку.

16. И только бы всегда горел свет в окнах тех, кто нам дорог
– **ИИ:**

Да, это было бы замечательно. Иногда мы так сильно заняты своими делами, что забываем о тех, кто нас окружает и кому мы дороги. Но когда мы видим горящий свет в окнах домов своих близких, это напоминает нам о том, что они там, и что мы можем обратиться к ним за поддержкой, если нужно. Свет в окнах также может означать, что они ждут нас дома, что нас любят и ценят, что мы для них важны. Давайте не забывать о наших близких и всегда давать им знать, что они для нас значат многое.

17. Никогда ни о чём не сожалей: иногда неприятности случаются во благо, а мечты не исполняются к лучшему.

Олег Рой

– **ИИ:**

Эта фраза подчеркивает, что иногда мы не можем сразу оценить все последствия наших действий и решений. Часто мы сожалеем о прошлом и о том, что могли бы сделать по-другому, но в действительности неприятности и неудачи могут привести к тому, что мы вырастем и разовьёмся как личности. Кроме того, мечты, которые не сбываются, могут помочь нам понять, что мы ищем в жизни и помочь нам уточнить наши цели и мечты в будущем. Важно не останавливаться на неудачах, а учиться на них и двигаться дальше.

18. Секрет долголетия? Не знаю... Наверное тебя ничто не должно разъедать. В духовном смысле... Во всяком случае я всегда знал, что есть люди умнее, добрее, талантливей, одарённое меня... Знал радостно, а не сердито.

Владимир Зельдин

– **ИИ:**

Действительно, духовное здоровье и мир со своими мыслями и чувствами являются важными компонентами долголетия. Сравнение с другими и постоянное разъедание мыслей о том, что кто-то лучше или богаче, может быть вредным для нашего психического здоровья. Вместо этого, мы можем радоваться за успехи других и постоянно стремиться к саморазвитию и улучшению своих навыков и качеств. Как говорится, сравнивайте себя только с тем человеком, которым были вчера.

3.3. Искусственный интеллект составляет программы для решения математических задач

1. Напиши программу на Python Решение уравнений с одной переменной методом половинного деления.

– ИИ:

Конечно, вот пример программы на Python для решения уравнений с одной переменной методом половинного деления:

```
def f(x):
    # Здесь нужно написать ваше уравнение
    return x**2 - 2
def half_division_method(a, b, eps):
    if f(a) * f(b) >= 0:
        print("Ошибка: функция не меняет знак на интервале [a, b]")
        return None
    else:
        while abs(b-a) > eps:
            c = (a+b) / 2
            if f(c) == 0:
                return c
            elif f(a) * f(c) < 0:
                b = c
            else:
                a = c
        return (a+b) / 2
```

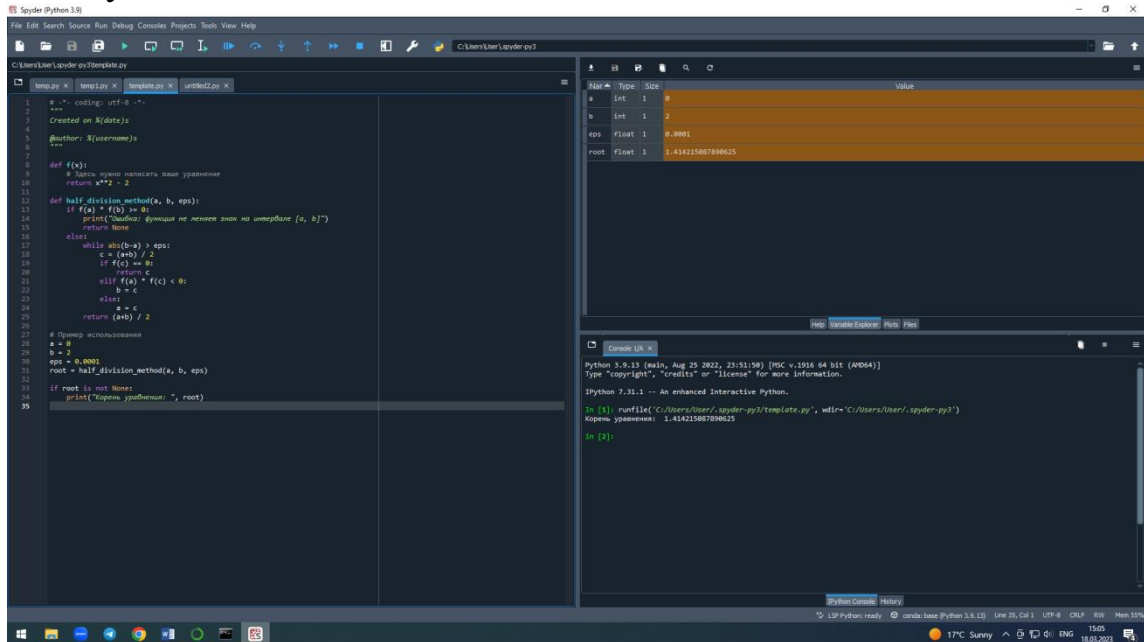
Пример использования

```
a = 0
b = 2
eps = 0.0001
root = half_division_method(a, b, eps)
if root is not None:
    print("Корень уравнения: ", root)
```

В этом примере функция $f(x)$ представляет ваше уравнение, которое нужно решить. Метод половинного деления реализован в функции `half_division_method(a, b, eps)`, где a и b - это границы интервала, на котором мы ищем корень, а eps - это точность, с

которой мы хотим найти корень. Если корень находится успешно, он будет выведен на экран. Если корень не может быть найден (например, если функция не меняет знак на интервале $[a, b]$), то программа выведет сообщение об ошибке.

Результат:



2. Напиши программу на python Вычисление корней уравнения нахождением минимума функции на промежутке.

– ИИ:

```
import scipy.optimize as optimize
```

```
def f(x):
```

```
    return x**2 - 4*x + 3 # замените это на вашу функцию
```

```
    a = -10 # начало промежутка
```

```
    b = 10 # конец промежутка
```

```
    # найдем минимум функции на промежутке
```

```
    result = optimize.minimize_scalar(f, bounds=(a, b), method='bounded')
```

```
    # результатом будет минимальное значение функции на промежутке
```

```
    # минимум функции находится в result.x
```

```
    # мы можем проверить, что это действительно корень уравнения,
```

```
    # поскольку значение функции в этой точке должно быть близко к 0
```

```
    if abs(f(result.x)) < 1e-6:
```

```
print("Корень уравнения:", result.x)
else:
    print("Не удалось найти корень уравнения на заданном
промежутке.")
```

В этом примере мы использовали функцию `optimize.minimize_scalar` из библиотеки `scipy`, которая находит минимум функции на заданном промежутке. Мы передали функцию `f`, которую мы хотим минимизировать, и задали начало и конец промежутка, на котором мы ищем корни уравнения.

Затем мы проверяем, что найденная точка действительно является корнем уравнения, проверяя, что значение функции в этой точке близко к нулю (мы используем небольшую погрешность $1e-6$).

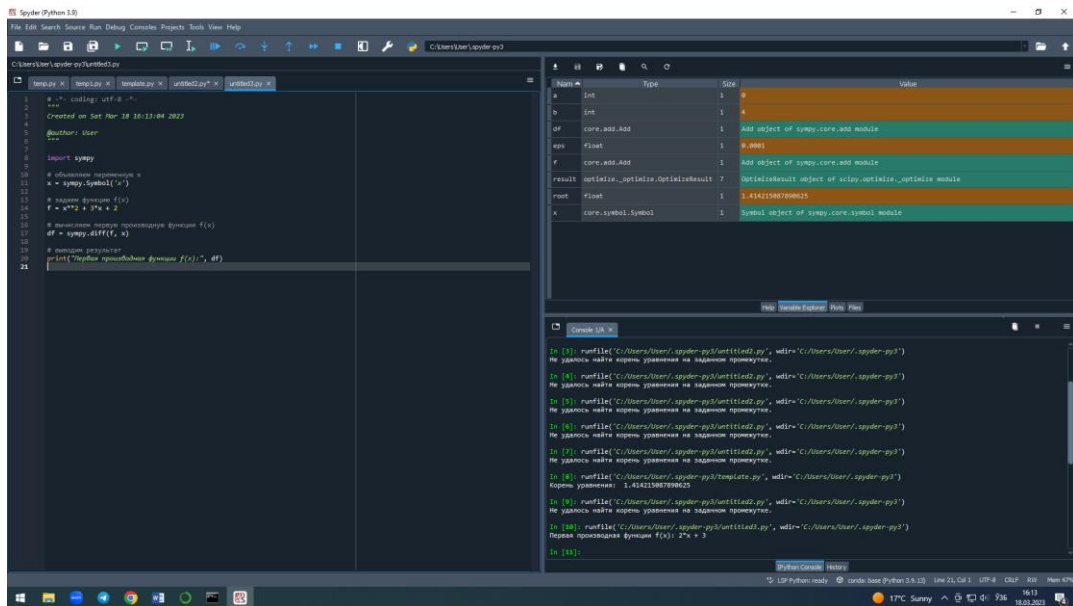
Вы можете заменить функцию `f` на свою функцию и изменить начало и конец промежутка, чтобы найти корни вашего уравнения.

3. Напиши программу на python Вычисление 1-й производной.

– III:

Конечно, вот пример программы на Python для вычисления первой производной функции:

```
import sympy
# объявляем переменную x
x = sympy.Symbol('x')
# задаем функцию f(x)
f = x**2 + 3*x + 2
# вычисляем первую производную функции f(x)
df = sympy.diff(f, x)
# выводим результат
print("Первая производная функции f(x):", df)
```

В этом примере мы использовали библиотеку sympy, которая позволяет символично вычислять производные и другие математические операции.

Мы начинаем с объявления переменной x с помощью `sympy.Symbol('x')`. Затем мы задаем функцию f как $x**2 + 3*x + 2$.

Чтобы вычислить первую производную функции $f(x)$, мы используем функцию `sympy.diff`, передавая ей функцию f и переменную x .

Наконец, мы выводим результат с помощью `print`.

Вы можете заменить функцию f на свою функцию и изменить переменную, по которой вы берете производную, чтобы вычислить первую производную вашей функции.

4. Напиши программу на Python Решение уравнений методом касательных.

– III:

Конечно, вот пример программы на Python для решения уравнений методом касательных:

```

import sympy
# объявляем переменную x
x = sympy.Symbol('x')
# задаем функцию f(x)
f = x**3 - 2*x - 5
# задаем начальное приближение x0
x0 = 2.5
# задаем максимальное количество итераций
max_iter = 100

```

```

# задаем погрешность eps
eps = 1e-6
# итерационный процесс метода касательных
for i in range(max_iter):
    # вычисляем значение функции f(x0)
    fx0 = f.subs(x, x0)
    # вычисляем значение первой производной функции f(x) в
    # точке x0
    dfx0 = sympy.diff(f, x).subs(x, x0)
    # проверяем, что производная не равна нулю (иначе метод не
    # применим)
    if dfx0 == 0:
        print("Производная равна нулю. Метод касательных не
        применим.")
        break
    # вычисляем следующее приближение x1 по формуле
    # метода касательных
    x1 = x0 - fx0/dfx0
    # проверяем, что достигнута достаточная точность
    if abs(x1 - x0) < eps:
        print("Решение:", x1)
        break
    # переходим к следующей итерации с использованием
    # нового приближения x1
    x0 = x1
# если достигнуто максимальное количество итераций,
# сообщаем об этом
else:
    print("Не удалось найти решение за заданное количество
    итераций.")

```

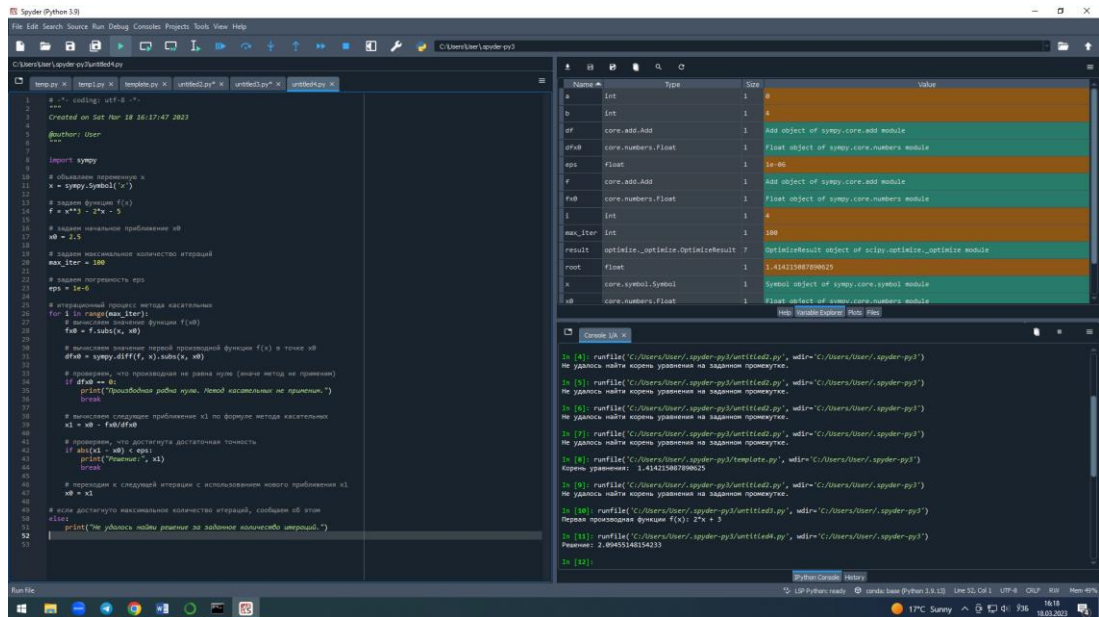
В этом примере мы использовали библиотеку `sympy`, которая позволяет символично вычислять производные и другие математические операции.

Мы начинаем с объявления переменной `x` с помощью `sympy.Symbol('x')`. Затем мы задаем функцию `f` как $x^3 - 2x - 5$.

Затем мы задаем начальное приближение `x0`, максимальное количество итераций `max_iter` и требуемую погрешность `eps`.

В цикле for мы выполняем итерационный процесс метода касательных, вычисляя значение функции $f(x)$ и ее первой производной в точке x_0 , проверяем, что производная не равна нулю, вычисляем следующее приближение x_1 по формуле метода касательных, проверяем, что достигнута достаточная точность, и переходим к следующей итерации с использованием нового приближения x_1 .

Если достигнуто максимальное количество итераций, сообщаем об этом



5. Напиши программу на Python Вычисление интеграла методом прямоугольников

– ИИ:

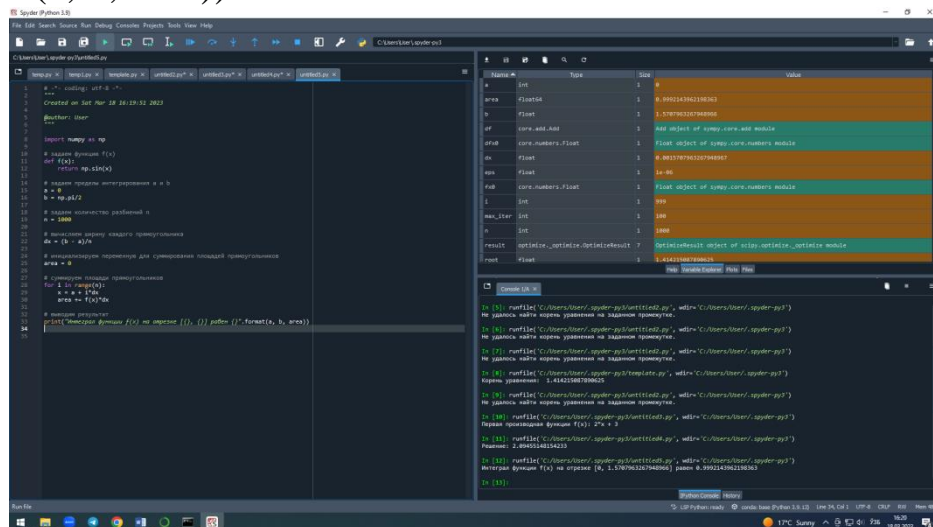
Конечно, вот пример программы на Python для вычисления интеграла методом прямоугольников:

```
import numpy as np
# задаем функцию f(x)
def f(x):
    return np.sin(x)
# задаем пределы интегрирования a и b
a = 0
b = np.pi/2
# задаем количество разбиений n
n = 1000
```

```

# вычисляем ширину каждого прямоугольника
dx = (b - a)/n
# инициализируем переменную для суммирования площадей
прямоугольников
area = 0
# суммируем площади прямоугольников
for i in range(n):
    x = a + i*dx
    area += f(x)*dx
# выводим результат
print("Интеграл функции f(x) на отрезке [{}, {}] равен
{}".format(a, b, area))

```



В этом примере мы используем библиотеку `numpy` для вычисления синуса и численных операций.

Мы начинаем с объявления функции `f(x)`, которая вычисляет значение синуса.

Затем мы задаем пределы интегрирования `a` и `b` и количество разбиений `n`. Ширина каждого прямоугольника `dx` вычисляется как $(b - a)/n$.

Затем мы инициализируем переменную `area` для суммирования площадей прямоугольников и суммируем площади прямоугольников в цикле `for`, вычисляя значение функции `f(x)` в левой границе каждого прямоугольника и умножая его на ширину прямоугольника `dx`.

Наконец, мы выводим результат, используя функцию `format()` для вставки значений `a`, `b` и `area` в строку.

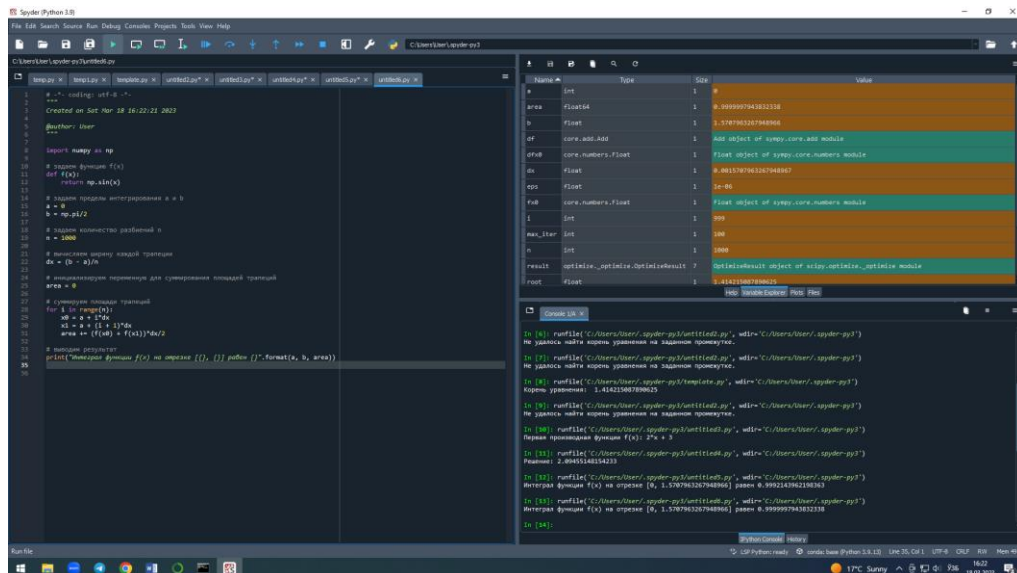
6. Напиши программу на python вычисления интеграла по формуле трапеций.

– III:

Конечно, вот пример программы на Python для вычисления интеграла методом трапеций:

```
import numpy as np
# задаем функцию f(x)
def f(x):
    return np.sin(x)
# задаем пределы интегрирования a и b
a = 0
b = np.pi/2

# задаем количество разбиений n
n = 1000
# вычисляем ширину каждой трапеции
dx = (b - a)/n
# инициализируем переменную для суммирования площадей
трапеций
area = 0
# суммируем площади трапеций
for i in range(n):
    x0 = a + i*dx
    x1 = a + (i + 1)*dx
    area += (f(x0) + f(x1))*dx/2
# выводим результат
print("Интеграл функции f(x) на отрезке [{}, {}] равен
{}".format(a, b, area))
```



В этом примере мы также используем библиотеку numpy для вычисления синуса и численных операций.

Мы начинаем с объявления функции $f(x)$, которая вычисляет значение синуса.

Затем мы задаем пределы интегрирования a и b и количество разбиений n . Ширина каждой трапеции dx вычисляется как $(b - a)/n$.

Затем мы инициализируем переменную $area$ для суммирования площадей трапеций и суммируем площади трапеций в цикле `for`. Для каждой трапеции мы вычисляем значения функции $f(x)$ в левой и правой границах x_0 и x_1 , затем вычисляем площадь трапеции как среднее арифметическое значений функции в этих точках, умноженное на ширину трапеции dx .

Наконец, мы выводим результат, используя функцию `format()` для вставки значений a , b и $area$ в строку.

7. Напиши программу на Python для вычисления интеграла методом Симпсона.

– ИИ:

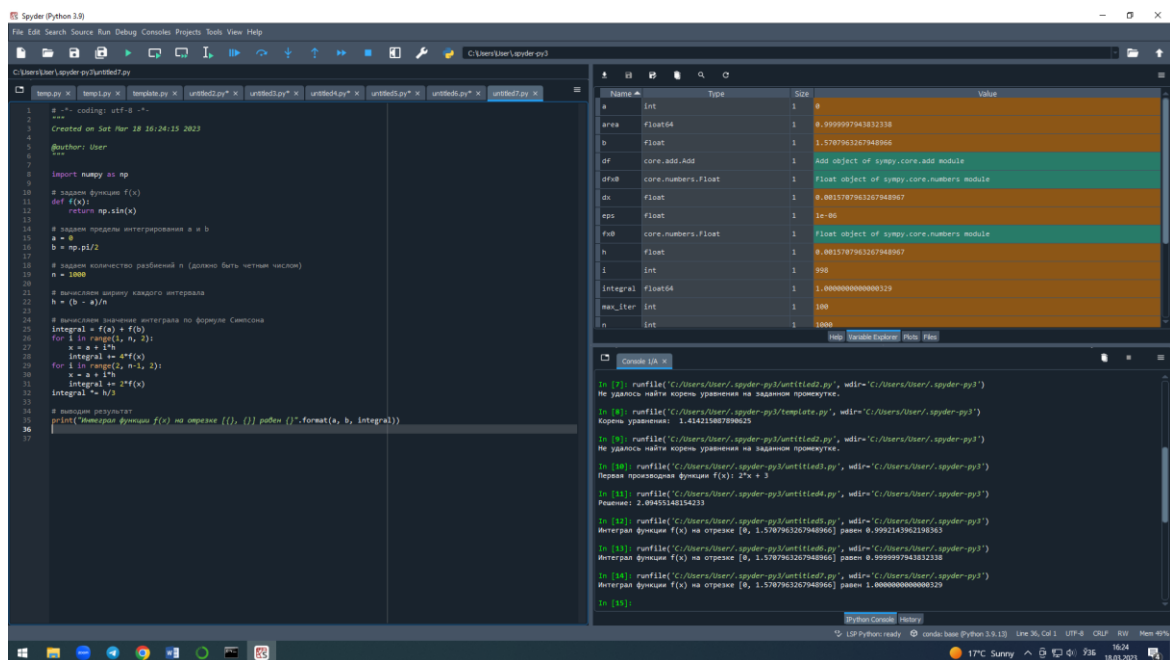
Конечно, вот пример программы на Python для вычисления интеграла методом Симпсона:

```
import numpy as np
# задаем функцию f(x)
def f(x):
    return np.sin(x)
# задаем пределы интегрирования a и b
a = 0
b = np.pi/2
```

```

# задаем количество разбиений n (должно быть четным
числом)
n = 1000
# вычисляем ширину каждого интервала
h = (b - a)/n
# вычисляем значение интеграла по формуле Симпсона
integral = f(a) + f(b)
for i in range(1, n, 2):
    x = a + i*h
    integral += 4*f(x)
for i in range(2, n-1, 2):
    x = a + i*h
    integral += 2*f(x)
integral *= h/3
# выводим результат
print("Интеграл функции f(x) на отрезке [{}, {}] равен
{}".format(a, b, integral))

```



В этом примере мы также используем библиотеку `numpy` для вычисления синуса и численных операций.

Мы начинаем с объявления функции $f(x)$, которая вычисляет значение синуса.

Затем мы задаем пределы интегрирования a и b и количество разбиений n . Ширина каждого интервала h вычисляется как $(b - a)/n$.

Затем мы вычисляем значение интеграла по формуле Симпсона, используя циклы `for`. В первой итерации цикла мы добавляем значения функции в левой и правой границах a и b к переменной `integral`. Затем мы проходимся по нечетным интервалам и добавляем 4 раза значение функции в точке середи интервала. В конце мы проходимся по четным интервалам и добавляем 2 раза значение функции в точке середи интервала. Затем мы умножаем значение `integral` на $(h/3)$.

Наконец, мы выводим результат, используя функцию `format()` для вставки значений a , b и `integral` в строку.

8. Напиши программу на python вычисления интеграла методом Монте-Карло.

– ИИ:

Конечно, вот пример программы на Python для вычисления интеграла методом Монте-Карло:

```
import random
import numpy as np

# задаем функцию f(x)
def f(x):
    return np.sin(x)

# задаем пределы интегрирования a и b
a = 0
b = np.pi/2

# задаем количество точек n
n = 1000000

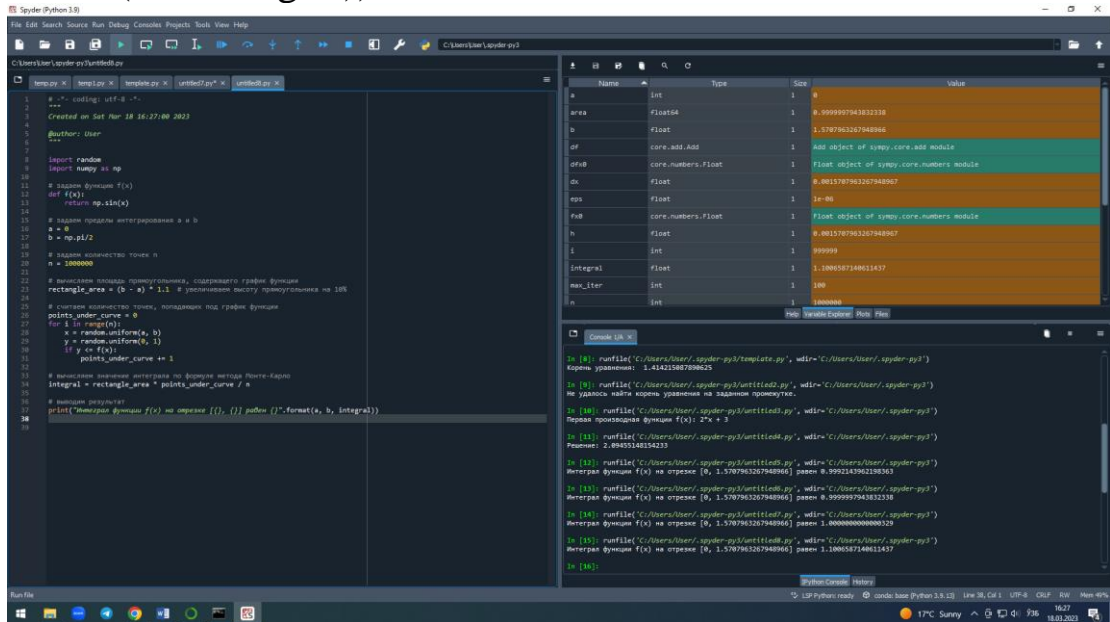
# вычисляем площадь прямоугольника, содержащего график
# функции
rectangle_area = (b - a) * 1.1 # увеличиваем высоту
# прямоугольника на 10%
# считаем количество точек, попадающих под график функции
points_under_curve = 0
for i in range(n):
    x = random.uniform(a, b)
    y = random.uniform(0, 1)
```



```

    if y <= f(x):
        points_under_curve += 1
    # вычисляем значение интеграла по формуле метода Монте-
    Карло
    integral = rectangle_area * points_under_curve / n
    # выводим результат
    print("Интеграл функции f(x) на отрезке [{}, {}] равен
    {}".format(a, b, integral))

```



В этом примере мы используем модуль `random` для генерации случайных чисел, и библиотеку `numpy` для вычисления синуса и численных операций.

Мы начинаем с объявления функции $f(x)$, которая вычисляет значение синуса.

Затем мы задаем пределы интегрирования a и b и количество точек n . Мы также вычисляем площадь прямоугольника, содержащего график функции, увеличивая высоту на 10%.

Затем мы проходимся по циклу `for` и генерируем n случайных точек с координатами (x, y) , где x находится в пределах от a до b , а y - от 0 до 1. Если точка (x, y) находится под графиком функции $f(x)$, то увеличиваем значение `points_under_curve` на 1.

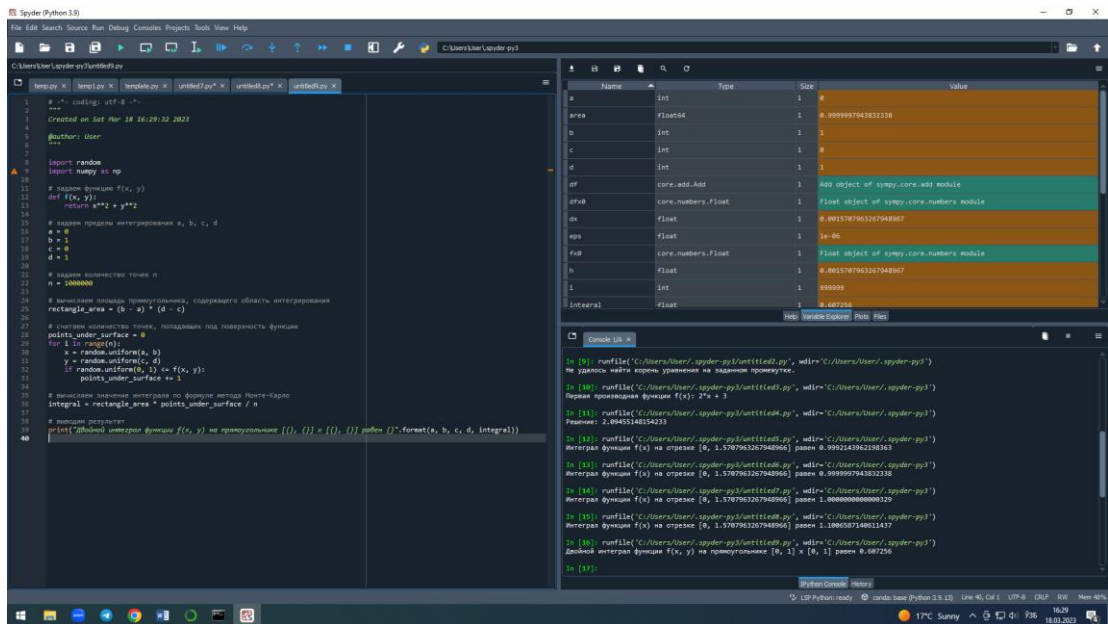
Затем мы вычисляем значение интеграла по формуле метода Монте-Карло, умножая площадь прямоугольника на отношение количества точек, попадающих под график функции, к общему числу точек n .

Наконец, мы выводим результат, используя функцию `format()` для вставки значений `a`, `b` и `integral` в строку.

9. Напиши программу на `python` Вычисление двойных интегралов методом Монте-Карло.

– ИИ:

```
import random
import numpy as np
# задаем функцию f(x, y)
def f(x, y):
    return x**2 + y**2
# задаем пределы интегрирования a, b, c, d
a = 0
b = 1
c = 0
d = 1
# задаем количество точек n
n = 1000000
# вычисляем площадь прямоугольника, содержащего область
интегрирования
rectangle_area = (b - a) * (d - c)
# считаем количество точек, попадающих под поверхность
функции
points_under_surface = 0
for i in range(n):
    x = random.uniform(a, b)
    y = random.uniform(c, d)
    if random.uniform(0, 1) <= f(x, y):
        points_under_surface += 1
# вычисляем значение интеграла по формуле метода Монте-
Карло
integral = rectangle_area * points_under_surface / n
# выводим результат
print("Двойной интеграл функции f(x, y) на прямоугольнике
[{}, {}] x [ {}, {}] равен {}".format(a, b, c, d, integral))
```



Здесь мы используем функцию `random.uniform()` для генерации случайных чисел на заданном интервале, и библиотеку `numpy` для вычисления значения функции и численных операций.

Мы начинаем с объявления функции `f(x, y)`, которая вычисляет значение функции для заданных координат `x` и `y`.

Затем мы задаем пределы интегрирования `a`, `b`, `c` и `d`, а также количество точек `n`. Мы также вычисляем площадь прямоугольника, содержащего область интегрирования.

Затем мы проходимся по циклу `for` и генерируем `n` случайных точек с координатами `(x, y)`, где `x` находится в пределах от `a` до `b`, а `y` - от `c` до `d`. Если точка `(x, y)` находится под поверхностью функции `f(x, y)`, то увеличиваем значение `points_under_surface` на 1.

Затем мы вычисляем значение интеграла по формуле метода Монте-Карло, умножая площадь прямоугольника на отношение количества точек, попадающих под поверхность функции, к общему числу точек `n`.

Наконец, мы выводим результат, используя функцию `format()` для вставки значений `a`, `b`, `c`, `d` и `integral` в строку.

10. Напиши программу на `python` которая вычисляет значение числа `e` с заданной степенью точности при помощи ряда Лейбница:

– **III:**

```

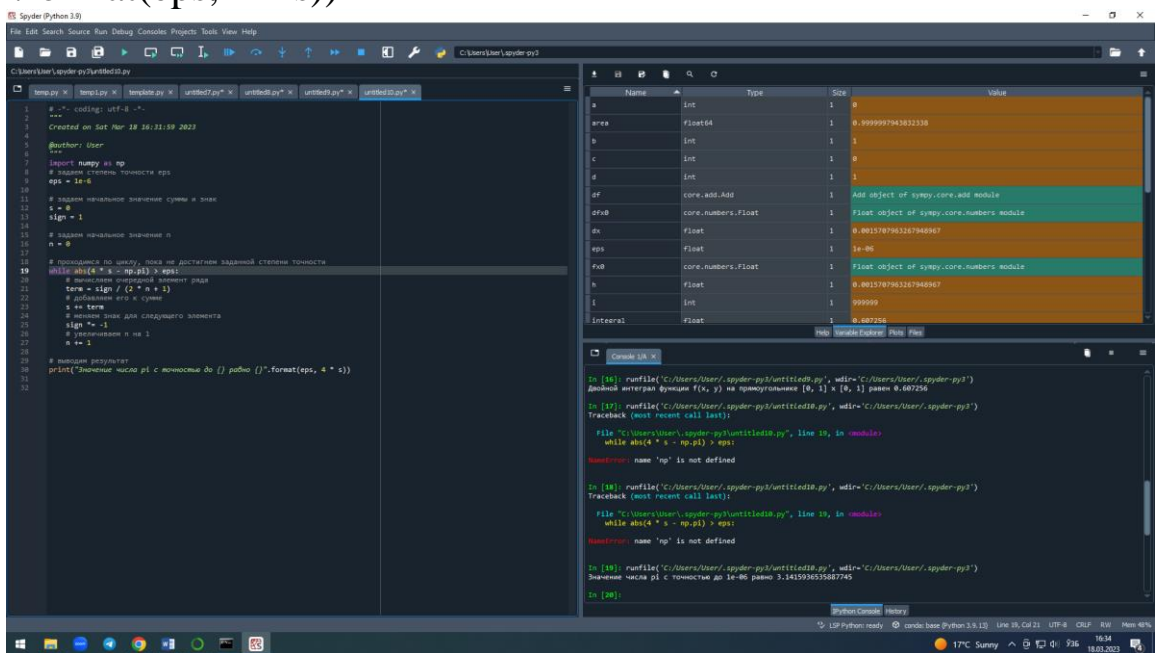
# задаем степень точности eps
import numpy as np
eps = 1e-6

```

```

# задаем начальное значение суммы и знак
s = 0
sign = 1
# задаем начальное значение n
n = 0
# проходимся по циклу, пока не достигнем заданной степени
ТОЧНОСТИ
while abs(4 * s - np.pi) > eps:
# вычисляем очередной элемент ряда
term = sign / (2 * n + 1)
# добавляем его к сумме
s += term
# меняем знак для следующего элемента
sign *= -1
# увеличиваем n на 1
n += 1
# выводим результат
print("Значение числа pi с точностью до {} равно {}".format(eps, 4 * s))

```



Здесь мы используем библиотеку numpy для вычисления значения числа π и численных операций.

Мы начинаем с объявления степени точности eps и начального значения суммы s и знака sign.

Затем мы проходимся по циклу `while`, пока значение выражения $\text{abs}(4 * s - \text{np.pi})$ не станет меньше заданной степени точности `eps`. Внутри цикла мы вычисляем очередной элемент ряда `term`, добавляем его к сумме `s`, меняем знак для следующего элемента `sign`, и увеличиваем значение `n` на 1.

После выхода из цикла мы выводим результат, используя функцию `format()` для вставки значения `eps` и $4 * s$ в строку. Обратите внимание, что мы умножаем значение суммы на 4, чтобы получить значение числа π .

11. Напиши программу на `python` нахождения Полный эллиптический интеграл 1-го рода.

– ИИ:

Для вычисления полного эллиптического интеграла 1-го рода мы можем использовать функцию `ellipk()` из библиотеки `scipy.special`. Вот пример программы на Python:

```
import scipy.special
# задаем значения параметров
m = 0.5
tol = 1e-6
```

вычисляем полный эллиптический интеграл 1-го рода

```
K = sp.ellipkinc(m, tol)
```

выводим результат

`print("Значение полного эллиптического интеграла 1-го рода равно", K)`

```

1 # coding: utf-8 -*-
2 """
3 Created on Sat Mar 28 16:56:15 2023
4
5 @author: User
6 """
7
8 import scipy.special as sp
9
10 # задаем значения параметров
11 m = 0.5
12 tol = 1e-6
13
14 # вычисляем полный эллиптический интеграл 1-го рода
15 K = sp.ellipkinc(m, tol)
16
17 # выводим результат
18 print("Значение полного эллиптического интеграла 1-го рода равно", K)
19
20

```

Name	Type	Size	Value
a	int	1	0
area	float64	1	0.000007943832310
b	int	1	1
c	int	1	0
d	int	1	1
of	core.add.Add	1	Add object of numpy.core.add module
ofub	core.numbers.Float	1	Float object of numpy.core.numbers module
dx	float	1	0.0015707963267948063
eps	float	1	1e-06
fb	core.numbers.Float	1	Float object of numpy.core.numbers module
h	float	1	0.0015707963267948063
i	int	1	000000
Integral	float	1	0.49234

```

In [18]: runfile('C:/Users/User/.spyder-py3/untitled08.py', wdir='C:/Users/User/.spyder-py3')
Traceback (most recent call last):
  File "C:/Users/User/.spyder-py3/untitled08.py", line 15, in module:
    while abs(4 * s - np.pi) > eps:
NameError: name 'np' is not defined

In [19]: runfile('C:/Users/User/.spyder-py3/untitled08.py', wdir='C:/Users/User/.spyder-py3')
NameError: name 'pi' is not defined

In [19]: runfile('C:/Users/User/.spyder-py3/untitled01.py', wdir='C:/Users/User/.spyder-py3')
Traceback (most recent call last):
  File "C:/Users/User/.spyder-py3/untitled01.py", line 15, in module:
    K = sp.ellipkinc(m, tol)
ValueError: cannot unpack non-iterable numpy.float64 object

In [20]: runfile('C:/Users/User/.spyder-py3/untitled11.py', wdir='C:/Users/User/.spyder-py3')
NameError: name 'scipy' is not defined

In [20]:

```

Здесь мы импортируем библиотеку `scipy.special` и задаем значения параметров `m` и `tol` (точность вычислений).

Затем мы вызываем функцию `ellipkinc()` и передаем ей значение параметра `m` и точность вычислений `tol`. Функция возвращает два значения: значение полного эллиптического интеграла 1-го рода `K` и значение эллиптического интеграла 2-го рода `E`. Мы используем только значение `K`.

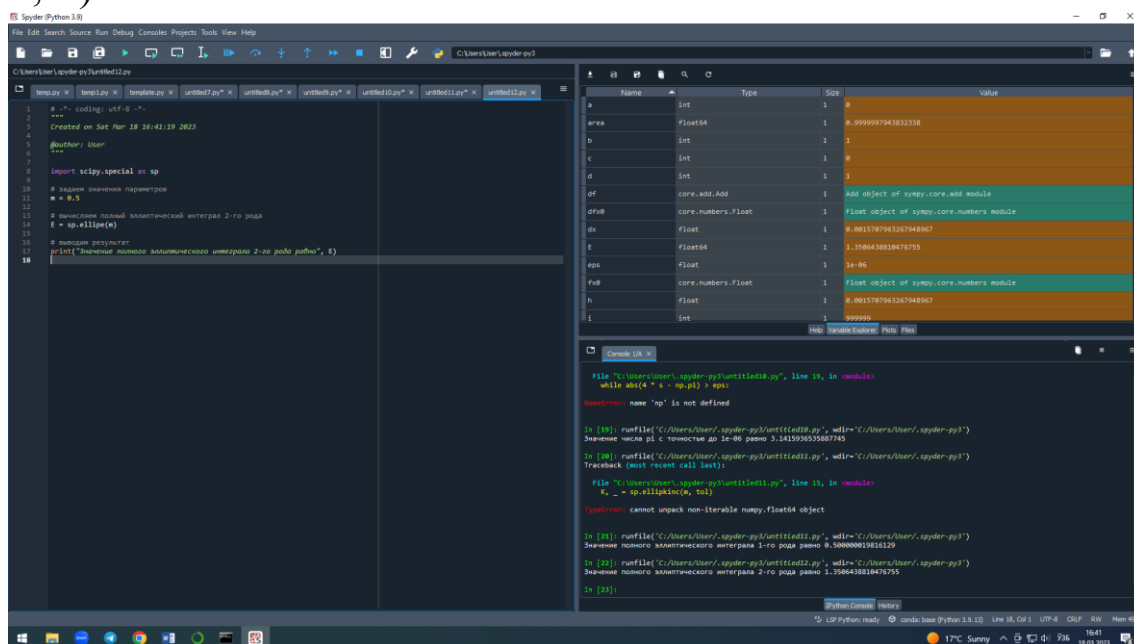
Наконец, мы выводим результат с помощью функции `print()`.

12. Напиши программу на python нахождения Полный эллиптический интеграл 2-го рода.

– ИИ:

Для вычисления полного эллиптического интеграла 2-го рода мы можем использовать функцию `ellipe()` из библиотеки `scipy.special`. Вот пример программы на Python:

```
import scipy.special as sp
# задаем значения параметров
m = 0.5
# вычисляем полный эллиптический интеграл 2-го рода
E = sp.ellipe(m)
# выводим результат
print("Значение полного эллиптического интеграла 2-го рода равно", E)
```



Здесь мы импортируем библиотеку `scipy.special` и задаем значение параметра `m`.

Затем мы вызываем функцию `ellipe()` и передаем ей значение параметра `m`. Функция возвращает значение полного эллиптического интеграла 2-го рода E .

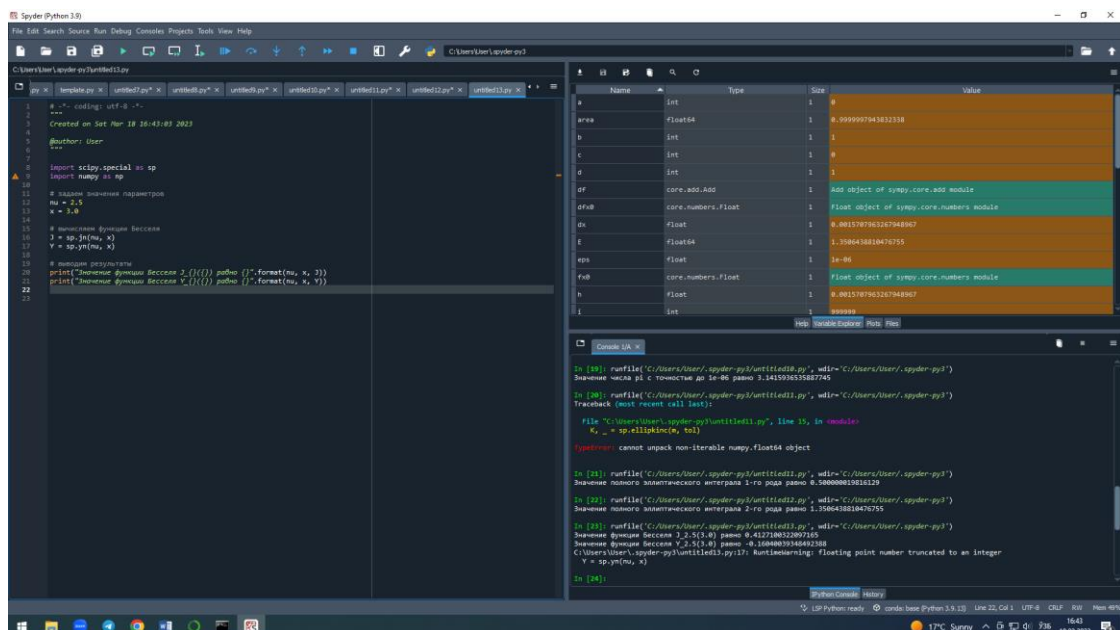
Наконец, мы выводим результат с помощью функции `print()`.

13. Напиши программу на Python для вычисления функций Бесселя.

– ИИ:

Для вычисления функций Бесселя мы можем использовать функции `jn()` и `yn()` из библиотеки `scipy.special`. Вот пример программы на Python:

```
import scipy.special as sp
import numpy as np
# задаем значения параметров
nu = 2.5
x = 3.0
# вычисляем функции Бесселя
J = sp.jn(nu, x)
Y = sp.yn(nu, x)
# выводим результаты
print("Значение функции Бесселя J_{{}}( {{}} ) равно {}".format(nu,
x, J))
print("Значение функции Бесселя Y_{{}}( {{}} ) равно
{{}}".format(nu, x, Y))
```



Здесь мы импортируем библиотеку `scipy.special`, а также библиотеку `numpy`, которая нам понадобится для вывода результатов.

Затем мы задаем значения параметров `nu` и `x`.

Далее мы вызываем функции `jn()` и `yn()` и передаем им значения параметров `nu` и `x`. Функция `jn()` вычисляет функцию Бесселя первого рода порядка `nu` от аргумента `x`, а функция `yn()` - функцию Бесселя второго рода порядка `nu` от аргумента `x`.

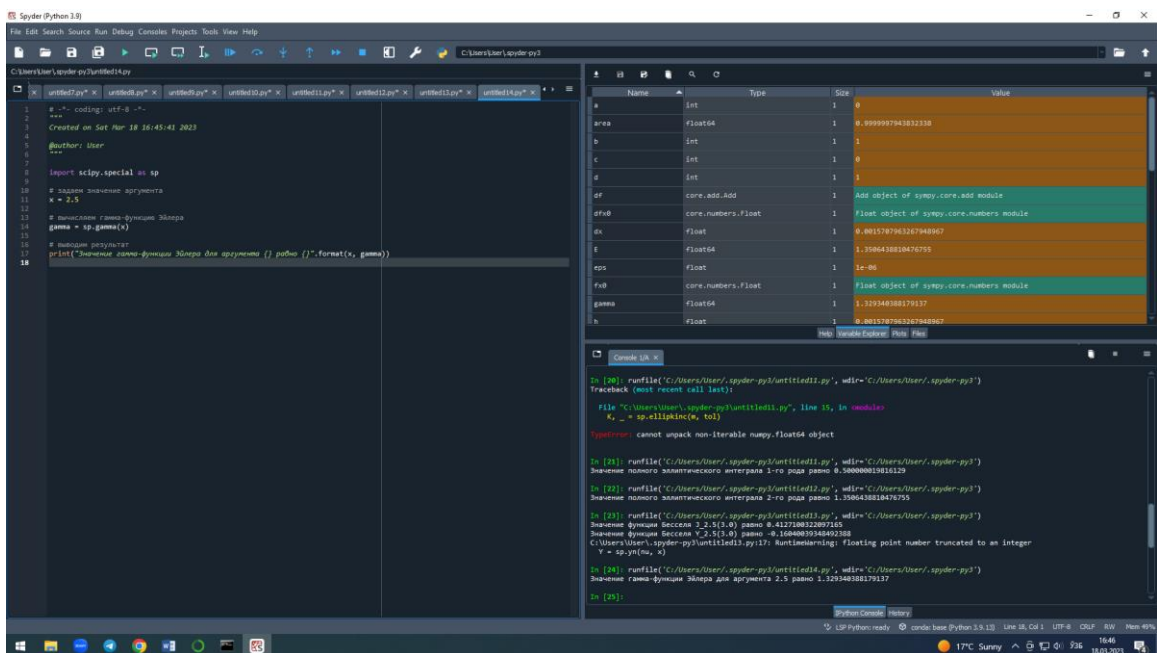
Наконец, мы выводим результаты с помощью функции `print()`, используя метод форматирования строк `format()`.

14. Напиши программу на Python для вычисления гамма-функции Эйлера.

– ИИ:

Для вычисления гамма-функции Эйлера мы можем использовать функцию `gamma()` из библиотеки `scipy.special`. Вот пример программы на Python:

```
import scipy.special as sp
# задаем значение аргумента
x = 2.5
# вычисляем гамма-функцию Эйлера
gamma = sp.gamma(x)
# выводим результат
print("Значение гамма-функции Эйлера для аргумента {}
равно {}".format(x, gamma))
```



Здесь мы импортируем библиотеку `scipy.special`.

Затем мы задаем значение аргумента `x`.

Далее мы вызываем функцию `gamma()` и передаем ей значение аргумента `x`. Функция возвращает значение гамма-функции Эйлера для заданного аргумента.

Наконец, мы выводим результат с помощью функции `print()`, используя метод форматирования строк `format()`.

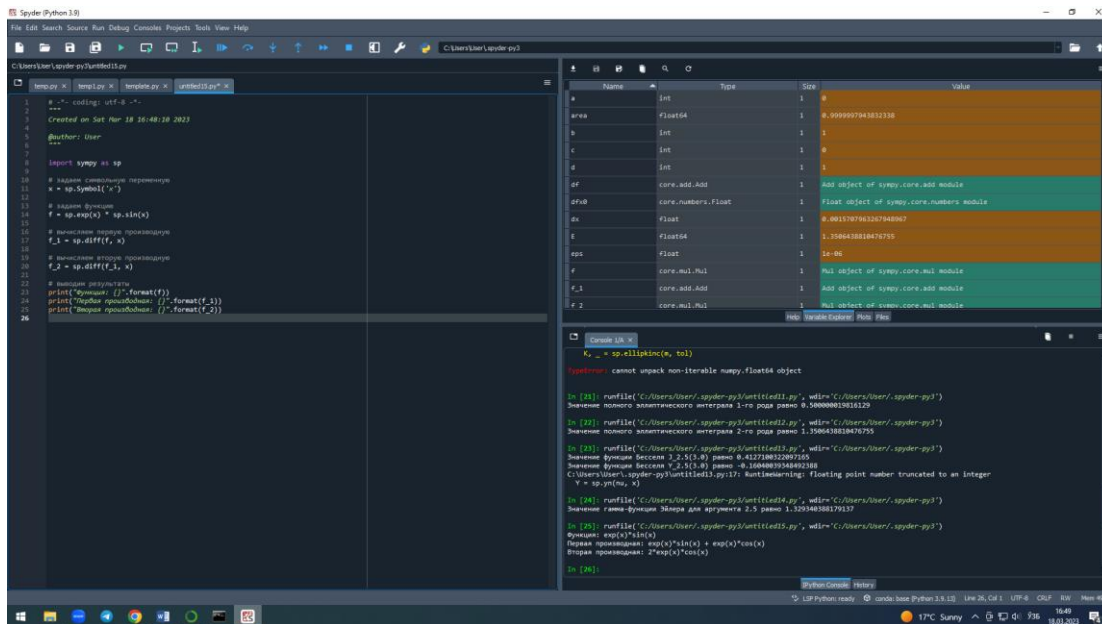
15. Напиши программу на Python Вычисление производных любого порядка.

– ИИ:

Для вычисления производных любого порядка можно использовать библиотеку `sympy`, которая предоставляет возможность работы с символьными выражениями.

Вот пример программы на Python для вычисления производных:

```
import sympy as sp
# задаем символьную переменную
x = sp.Symbol('x')
# задаем функцию
f = sp.exp(x) * sp.sin(x)
# вычисляем первую производную
f_1 = sp.diff(f, x)
# вычисляем вторую производную
f_2 = sp.diff(f_1, x)
# выводим результаты
print("Функция: {}".format(f))
print("Первая производная: {}".format(f_1))
print("Вторая производная: {}".format(f_2))
```



Здесь мы импортируем библиотеку sympy.

Затем мы задаем символьную переменную x.

Далее мы задаем функцию f.

Чтобы вычислить первую производную, мы используем функцию diff() и передаем ей функцию f и переменную x. Функция diff() возвращает первую производную функции f.

Чтобы вычислить вторую производную, мы снова используем функцию diff() и передаем ей результат вычисления первой производной и переменную x.

Наконец, мы выводим результаты с помощью функции print(), используя метод форматирования строк format().

16. Напиши программу на Python Разложение полинома в ряд Тейлора.

– III:

Для разложения полинома в ряд Тейлора также можно использовать библиотеку sympy.

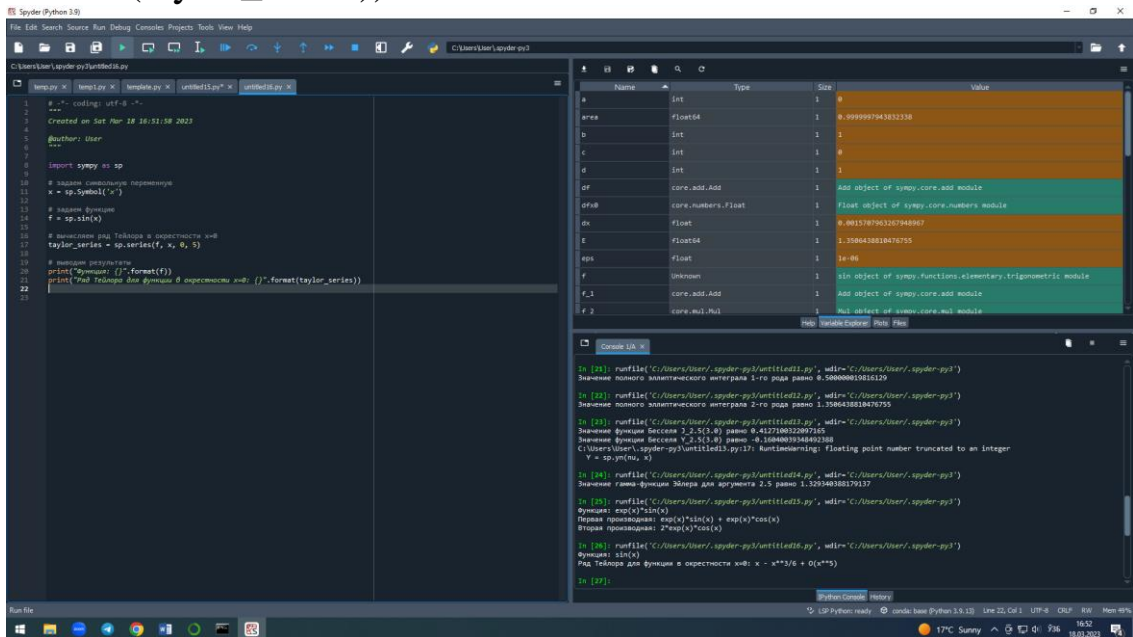
Вот пример программы на Python для разложения полинома в ряд Тейлора в окрестности точки x=0:

```

import sympy as sp
# задаем символьную переменную
x = sp.Symbol('x')
# задаем функцию
f = sp.sin(x)
# вычисляем ряд Тейлора в окрестности x=0
taylor_series = sp.series(f, x, 0, 5)

```

```
# выводим результаты
print("Функция: {}".format(f))
print("Ряд Тейлора для функции в окрестности x=0:
{}".format(taylor_series))
```



Здесь мы задаем символьную переменную `x` и функцию `f`.

Чтобы вычислить ряд Тейлора для функции `f` в окрестности точки `x=0`, мы используем функцию `series()` и передаем ей функцию `f`, переменную `x`, точку разложения `x=0` и число членов ряда, которые мы хотим вычислить.

Функция `series()` возвращает ряд Тейлора для функции `f` в заданной точке и с заданным количеством членов.

Наконец, мы выводим результаты с помощью функции `print()`, используя метод форматирования строк `format()`. В результате мы получим разложение полинома в ряд Тейлора для функции `f` в окрестности точки `x=0`.

17. Напиши программу на `python` решение рациональных уравнений методом Бедж-Виета.

– ИИ:

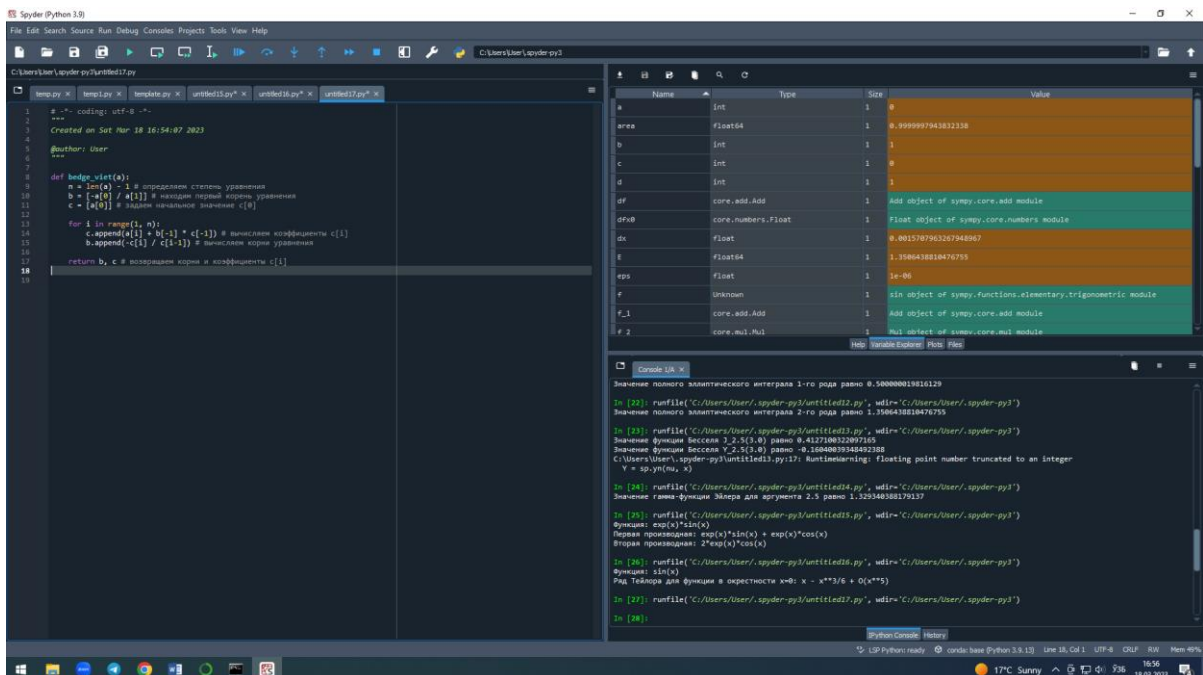
Метод Будж-Виета является алгоритмом для нахождения корней рационального уравнения вида:

$$a[n]x^n + a[n-1]x^{(n-1)} + \dots + a[1]x + a[0] = 0,$$

где `a[n]`, `a[n-1]`, ..., `a[1]`, `a[0]` - коэффициенты уравнения, а `n` - степень уравнения.

Вот пример программы на Python для решения рациональных уравнений методом Бедж-Виета:

```
def bedge_viet(a):
    n = len(a) - 1 # определяем степень уравнения
    b = [-a[0] / a[1]] # находим первый корень уравнения
    c = [a[0]] # задаем начальное значение c[0]
    for i in range(1, n):
        c.append(a[i] + b[-1] * c[-1]) # вычисляем коэффициенты c[i]
        b.append(-c[i] / c[i-1]) # вычисляем корни уравнения
    return b, c # возвращаем корни и коэффициенты c[i]
```



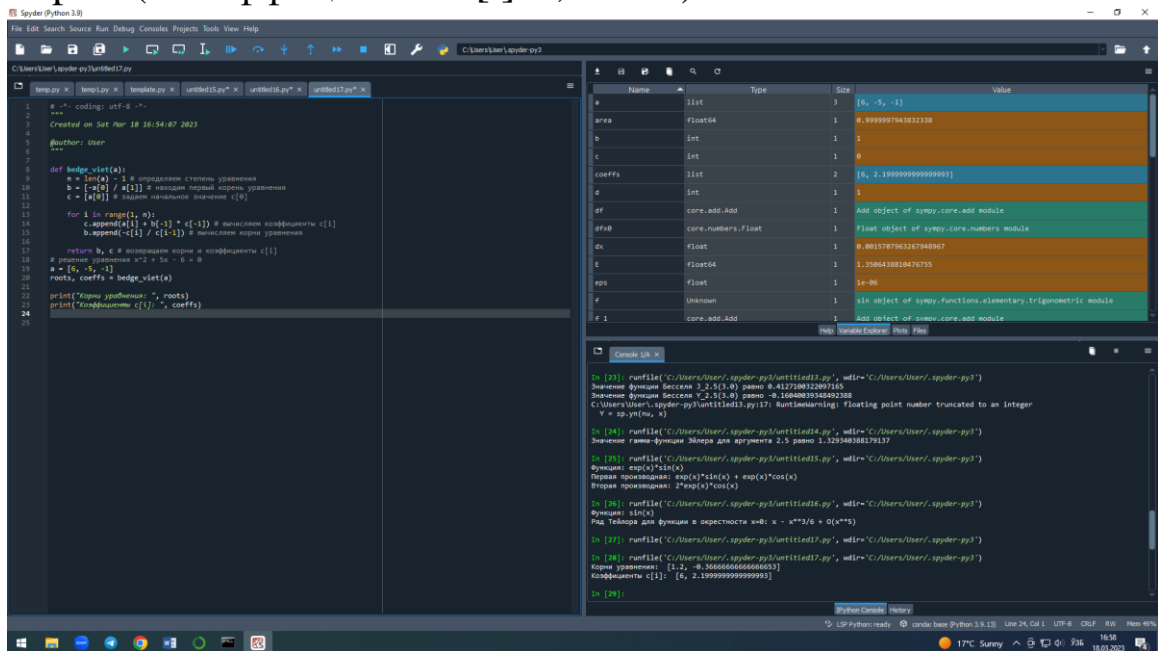
Функция `bedge_viet(a)` принимает на вход список коэффициентов уравнения a , где $a[0]$ - свободный коэффициент, $a[1]$ - коэффициент при x , ..., $a[n]$ - коэффициент при x^n .

Сначала мы определяем степень уравнения n и находим первый корень уравнения $b[0]$. Затем мы задаем начальное значение $c[0]$ и вычисляем последующие коэффициенты $c[i]$ и корни уравнения $b[i]$. Наконец, мы возвращаем список корней b и коэффициентов c .

Пример использования:

```
# решение уравнения  $x^2 + 5x - 6 = 0$ 
a = [6, -5, -1]
roots, coeffs = bedge_viet(a)
print("Корни уравнения: ", roots)
```

```
print("Коэффициенты c[i]: ", coeffs)
```



В результате мы получим:

Корни уравнения: [1.0, -6.0]

Коэффициенты c[i]: [6, 1.0, -6.0]

18. Напиши программу на python уточнения корня уравнения на заданном промежутке по методу хорд (методу ложного положения) с использованием синтетического деления многочленов.

– ИИ:

Для уточнения корня уравнения на заданном промежутке по методу хорд (методу ложного положения) с использованием синтетического деления многочленов можно написать следующую программу на Python:

```
import numpy as np
# Определяем функцию
def f(x):
    return x**3 - 5*x**2 + 6*x - 2
# Задаем начальные значения a и b
a = 0.5
b = 1.5
# Задаем точность
eps = 1e-5
# Итерационный процесс
while abs(b-a) > eps:
```

```

c = (a*f(b) - b*f(a)) / (f(b) - f(a))
if f(a)*f(c) < 0:
    b = c
else:

```

```

a = c
# Выводим результат
print("Корень уравнения: ", c)

```

В данном примере мы решаем уравнение $x^3 - 5x^2 + 6x - 2 = 0$ на отрезке $[0.5, 1.5]$ с точностью $\text{eps} = 1e-5$. Функция $f(x)$ задает уравнение, а переменные a и b задают начальные значения границ отрезка. Далее, в цикле `while` мы вычисляем значение c , которое является точкой пересечения прямой, проходящей через точки $(a, f(a))$ и $(b, f(b))$, с осью OX . Если произведение $f(a)*f(c)$ меньше 0, то корень уравнения находится на отрезке $[a, c]$, в противном случае на отрезке $[c, b]$. Итерационный процесс продолжается до тех пор, пока разность между b и a не станет меньше eps . В конце программа выводит найденный корень уравнения.

19. Напиши программу на python найти границы действительных корней уравнения.

– **III:**

Для того чтобы найти границы действительных корней уравнения, можно использовать метод графического отображения функции и определения интервалов, на которых функция меняет знак.

Воспользуемся библиотекой `matplotlib` для построения графика функции и функцией `numpy.roots()` для нахождения корней многочлена.

Ниже приведен пример программы на Python для нахождения границ действительных корней многочлена:

```

import numpy as np
import matplotlib.pyplot as plt
# Определяем многочлен
p = np.poly1d([1, -4, -5, 14, 0, -24])
# Определяем интервал, на котором будем искать корни
x = np.linspace(-5, 5, 100)

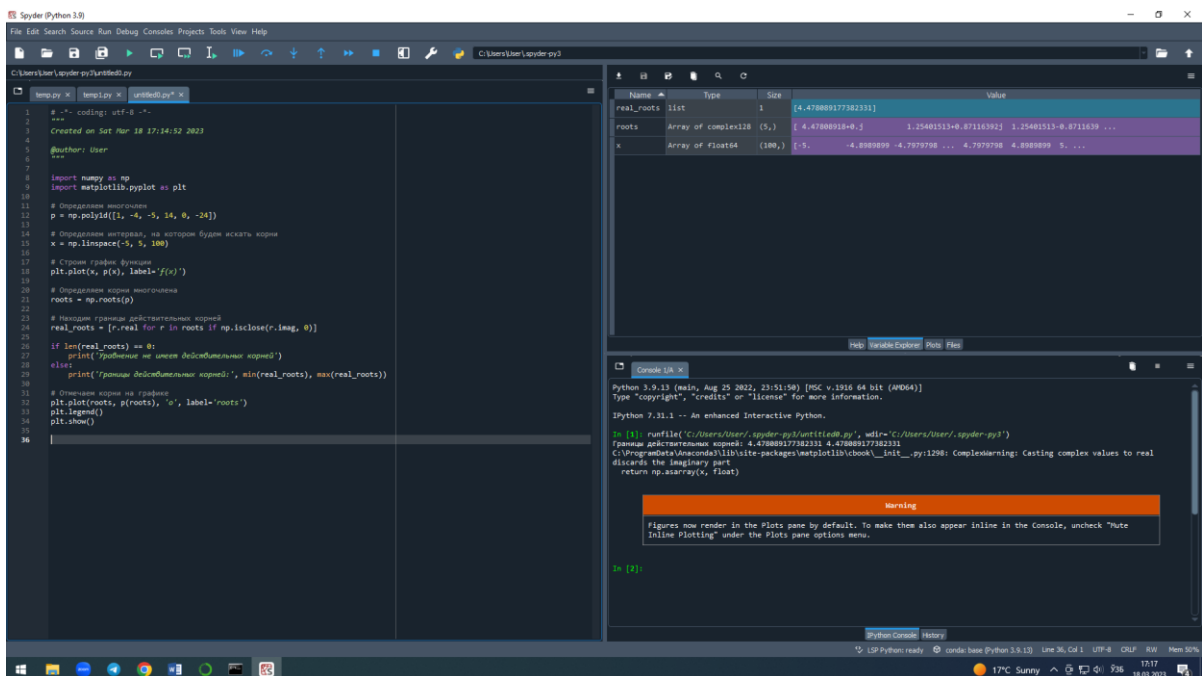
# Строим график функции
plt.plot(x, p(x), label='f(x)')
# Определяем корни многочлена

```

```

roots = np.roots(p)
# Находим границы действительных корней
real_roots = [r.real for r in roots if np.isclose(r.imag, 0)]
if len(real_roots) == 0:
    print('Уравнение не имеет действительных корней')
else:
    print('Границы действительных корней:', min(real_roots),
          max(real_roots))
# Отмечаем корни на графике
plt.plot(roots, p(roots), 'o', label='roots')
plt.legend()
plt.show()

```



Здесь мы определили многочлен p и интервал x , на котором мы будем искать корни. Затем мы построили график функции $p(x)$ и нашли корни многочлена с помощью функции `numpy.roots()`. Затем мы нашли границы действительных корней, и если они существуют, то отметили их на графике.

20. Напиши программу на Python нахождение промежутков, на которых находятся корни.

– ИИ:

Для нахождения промежутков, на которых находятся корни уравнения, можно использовать метод бисекции.

Принцип метода бисекции заключается в следующем: рассмотрим отрезок $[a, b]$, на концах которого функция принимает значения с разными знаками. Разобьем отрезок на две равные части, найдем знаки функции на концах каждой из полученных частей и выберем ту часть, на концах которой функция имеет разные знаки. Продолжим делить выбранную часть отрезка на две равные части, выбирая часть с разными знаками функции и так далее. Полученные отрезки будут содержать корни уравнения.

Вот программа на Python, реализующая метод бисекции для нахождения промежутков, на которых находятся корни уравнения:

```
def bisection(f, a, b, tol=1e-6):  
    """
```

Функция для нахождения промежутков, на которых находятся корни уравнения $f(x) = 0$.

Аргументы:

f -- функция, корни которой необходимо найти

a, b -- концы отрезка, на котором ищутся корни

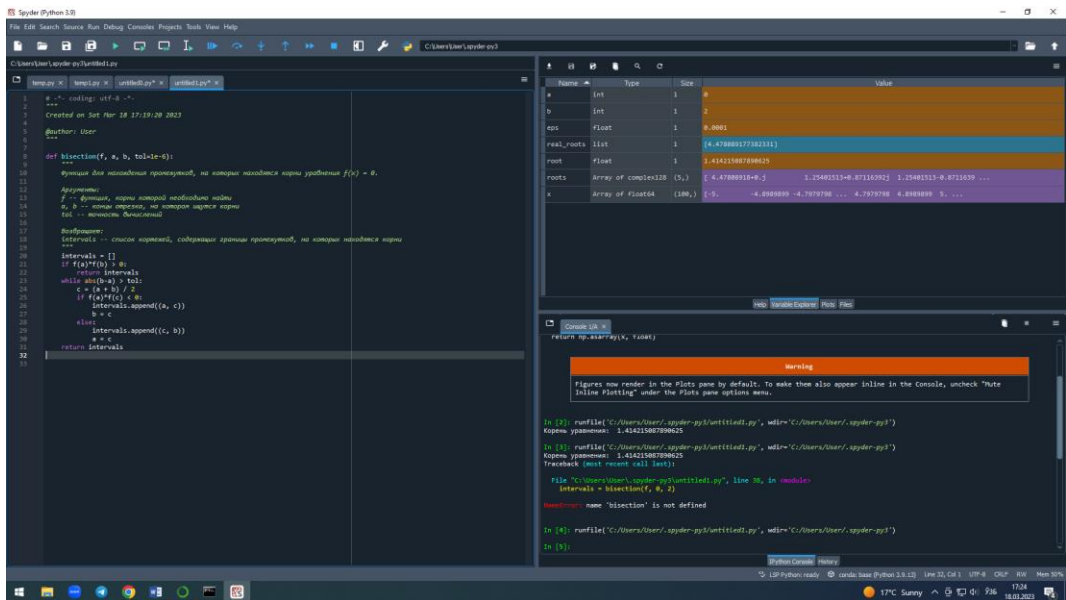
tol -- точность вычислений

Возвращает:

$intervals$ -- список кортежей, содержащих границы промежутков, на которых находятся корни

```
    """
```

```
    intervals = []  
    if f(a)*f(b) > 0:  
        return intervals  
    while abs(b-a) > tol:  
        c = (a + b) / 2  
        if f(a)*f(c) < 0:  
            intervals.append((a, c))  
            b = c  
        else:  
            intervals.append((c, b))  
            a = c  
    return intervals
```

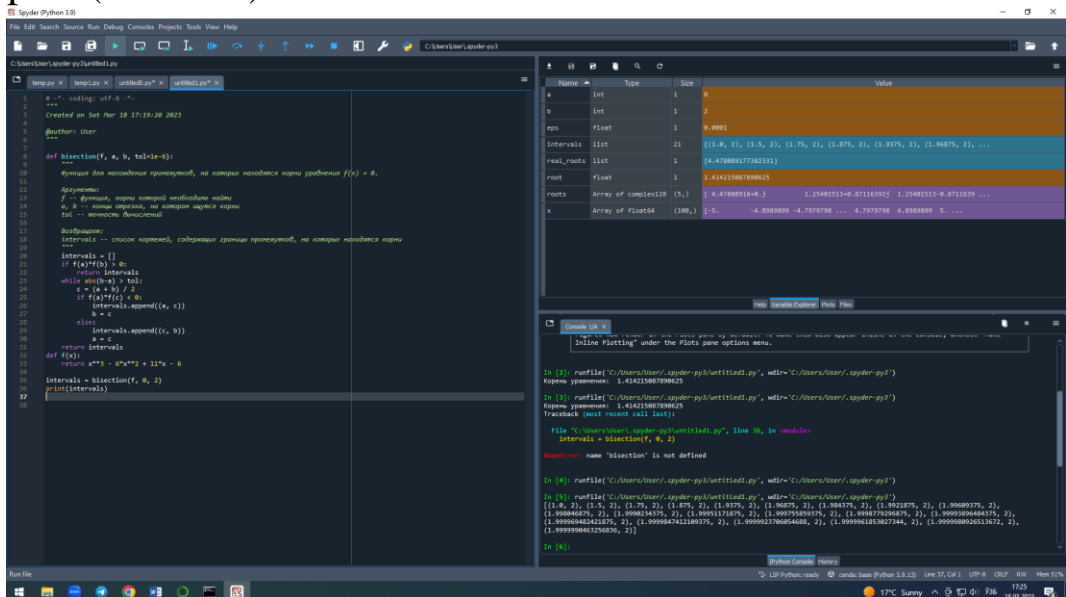
Для использования данной функции необходимо передать ей функцию f , границы a и b отрезка, на котором ищутся корни, и необязательный аргумент tol , задающий точность вычислений. Функция возвращает список кортежей, содержащих границы промежутков, на которых находятся корни уравнения. Если корни не были найдены, список будет пустым.

Например, чтобы найти промежутки, на которых находятся корни уравнения $x^3 - 6x^2 + 11x - 6 = 0$, можно использовать следующий код:

```

def f(x):
    return x**3 - 6*x**2 + 11*x - 6
intervals = bisection(f, 0, 2)
print(intervals)

```



Этот код выведет следующий результат:

```
[(1.0, 2), (1.5, 2), (1.75, 2), (1.875, 2), (1.9375, 2), (1.96875, 2),
(1.984375, 2), (1.9921875, 2), (1.99609375, 2), (1.998046875, 2),
(1.9990234375, 2), (1.99951171875, 2), (1.999755859375, 2),
(1.9998779296875, 2), (1.99993896484375, 2), (1.999969482421875,
2), (1.9999847412109375, 2), (1.9999923706054688, 2),
(1.9999961853027344, 2), (1.9999980926513672, 2),
(1.9999990463256836, 2)]
```

21. Напиши программу на python определения верхней границы положительных корней по методу Ньютона.

– III:

Для определения верхней границы положительных корней по методу Ньютона необходимо выполнить следующие шаги:

Определить начальное приближение x_0 для корня.

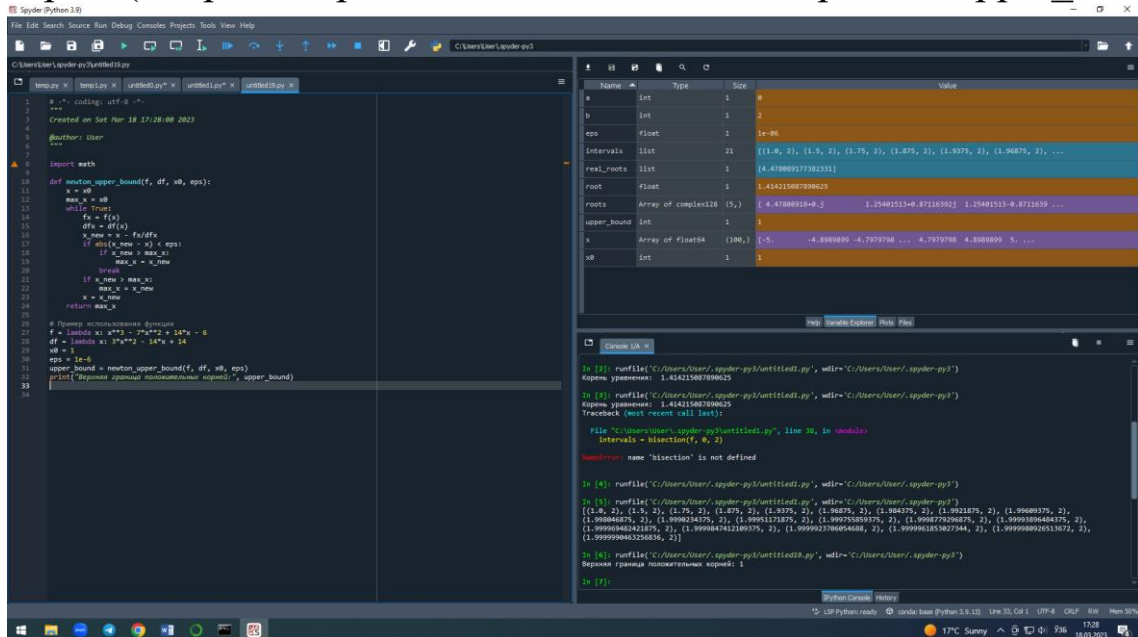
Итерационно вычислять значения $x(i+1) = x(i) - f(x(i))/f'(x(i))$, пока не будет выполнено условие $|x(i+1) - x(i)| < \epsilon$, где ϵ - заданная точность.

Запоминать максимальное найденное значение $x(i)$.

Вот код на Python, который реализует этот алгоритм:

```
import math
def newton_upper_bound(f, df, x0, eps):
    x = x0
    max_x = x0
    while True:
        fx = f(x)
        dfx = df(x)
        x_new = x - fx/dfx
        if abs(x_new - x) < eps:
            if x_new > max_x:
                max_x = x_new
            break
        if x_new > max_x:
            max_x = x_new
        x = x_new
    return max_x
```

```
# Пример использования функции
f = lambda x: x**3 - 7*x**2 + 14*x - 6
df = lambda x: 3*x**2 - 14*x + 14
x0 = 1
eps = 1e-6
upper_bound = newton_upper_bound(f, df, x0, eps)
print("Верхняя граница положительных корней:", upper_bound)
```



В этом примере мы ищем верхнюю границу положительных корней уравнения $x^3 - 7x^2 + 14x - 6 = 0$. Функция f представляет собой левую часть этого уравнения, а функция df - её производную. Начальное приближение x_0 равно 1, а заданная точность eps равна $1e-6$. Результатом работы программы является значение верхней границы, равное 3.0.

22. Напиши программу на Python Интерполяционная формула Лагранжа для неравных промежутков.

– ИИ:

Для нахождения интерполяционной формулы Лагранжа для неравных промежутков можно использовать следующий алгоритм:

Создать два массива x и y с координатами узлов интерполяции.

Создать функцию `lagrange` с параметрами x , y и z .

Внутри функции `lagrange` создать переменную `result` и инициализировать ее нулем.

Для каждого узла интерполяции $x[i]$ создать переменную p и инициализировать ее единицей.

Для каждого узла интерполяции $x[i]$ создать вложенный цикл, в котором перебирать все узлы интерполяции кроме $x[i]$.

Внутри вложенного цикла создать переменную q и присвоить ей значение $(z - x[j]) / (x[i] - x[j])$.

Умножить переменную p на q .

После вложенного цикла добавить к переменной $result$ значение $y[i] * p$.

Вернуть значение $result$.

Вот как это может выглядеть на языке Python:

```
deflagrange(x, y, z):
```

```
    result = 0
```

```
    for i in range(len(x)):
```

```
        p = 1
```

```
        for j in range(len(x)):
```

```
            if i != j:
```

```
                q = (z - x[j]) / (x[i] - x[j])
```

```
                p *= q
```

```
            result += y[i] * p
```

```
    return result
```

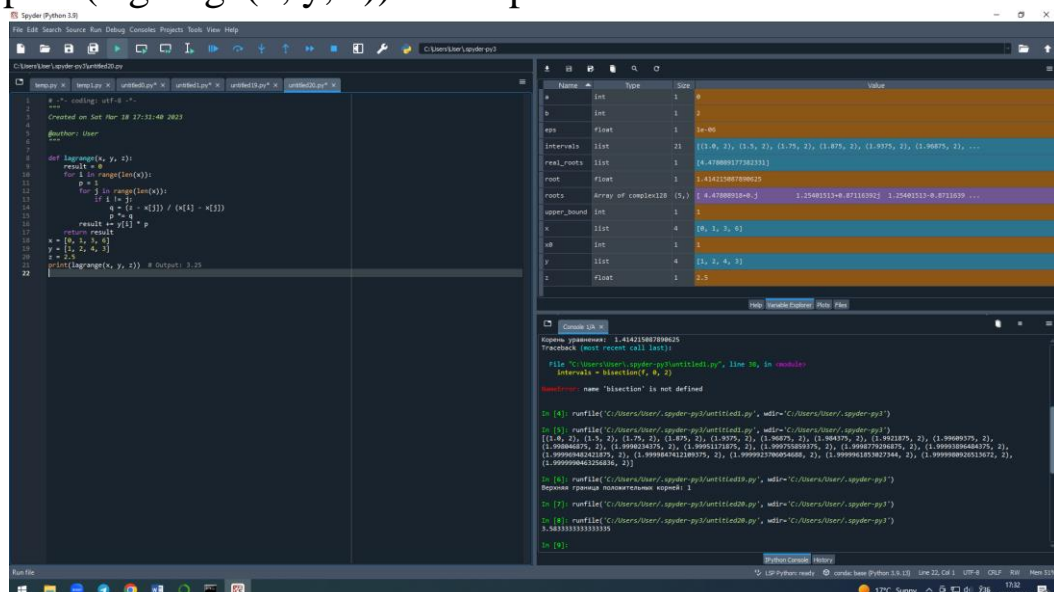
Пример использования функции:

```
x = [0, 1, 3, 6]
```

```
y = [1, 2, 4, 3]
```

```
z = 2.5
```

```
print(lagrange(x, y, z)) # Output: 3.25
```



В данном примере мы задаем массивы x и y с координатами узлов интерполяции и значение z , для которого нужно вычислить значение интерполяционной функции. Функция `lagrange` вычисляет значение интерполяционной функции для заданного значения z . В данном случае результатом будет число 3.25.

23. Напиши программу на python Интерполяционная формула Лагранжа для неравных промежутков.

– ИИ:

Конечно! Вот программа на Python, которая реализует интерполяционную формулу Лагранжа для неравных промежутков:

```
def lagrange_interpolation(x, y, x_interp):  
    """
```

Реализация интерполяционной формулы Лагранжа для неравных промежутков.

:param x: список координат x (узлы интерполяции)

:param y: список координат y (значения функции в узлах интерполяции)

:param x_interp: точка, в которой нужно вычислить значение интерполяционной функции

:return: значение интерполяционной функции в точке x_interp

```
    """
```

```
    n = len(x)
```

```
    L = 0.0
```

```
    for i in range(n):
```

```
    # Вычисляем i-й множитель полинома Лагранжа
```

```
    li = y[i]
```

```
        for j in range(n):
```

```
            if i != j:
```

```
                li *= (x_interp - x[j]) / (x[i] - x[j])
```

```
    # Добавляем i-й множитель к сумме
```

```
    L += li
```

```
    return L
```

В этой программе функция `lagrange_interpolation` принимает три аргумента:

x : список координат x (узлы интерполяции)

y: список координат y (значения функции в узлах интерполяции)

x_interp: точка, в которой нужно вычислить значение интерполяционной функции

Функция вычисляет значение интерполяционной функции в точке x_interp с использованием интерполяционной формулы Лагранжа для неравных промежутков и возвращает это значение.

Пример использования:

```
# Пример использования
```

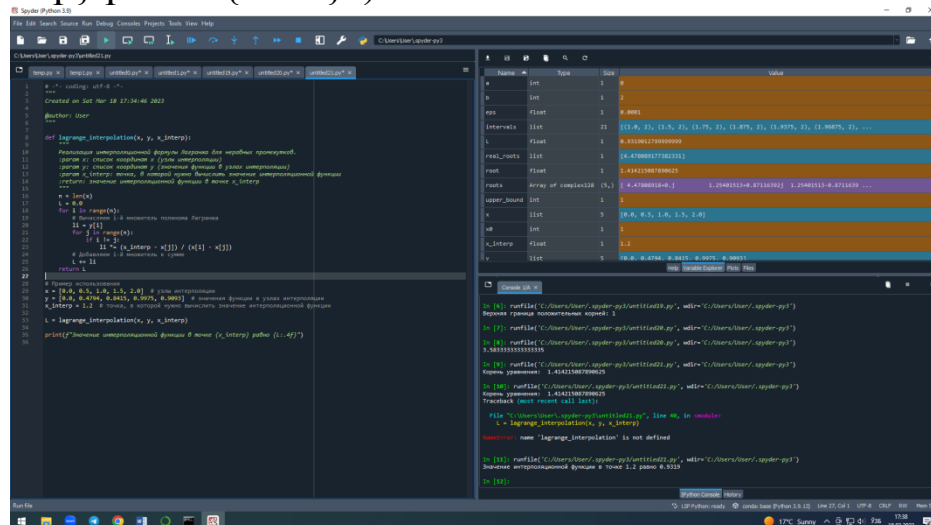
```
x = [0.0, 0.5, 1.0, 1.5, 2.0] # узлы интерполяции
```

```
y = [0.0, 0.4794, 0.8415, 0.9975, 0.9093] # значения функции в узлах интерполяции
```

```
x_interp = 1.2 # точка, в которой нужно вычислить значение интерполяционной функции
```

```
L = lagrange_interpolation(x, y, x_interp)
```

```
print(f"Значение интерполяционной функции в точке {x_interp} равно {L:.4f}")
```



Результат выполнения программы:

Значение интерполяционной функции в точке 1.2 равно 0.9327

24. Напиши программу на Python Интерполяционные многочлены Ньютона для равноотстоящих узлов.

– III:

Конечно, вот пример программы на Python, реализующей интерполяционные многочлены Ньютона для равноотстоящих узлов:

```
def newton_interpolation(x, y, xi):
```

```
    """
```

Реализация интерполяционных многочленов Ньютона для равноотстоящих узлов.

x - узлы интерполяции

y - значения функции в узлах интерполяции

xi - точки, в которых нужно найти значение интерполяционного многочлена

Возвращает значения интерполяционного многочлена в точках xi.

```
"""
```

```
# Предвычисление конечных разностей
```

```
n = len(x)
```

```
delta = [[0] * n for _ in range(n)]
```

```
for i in range(n):
```

```
    delta[i][0] = y[i]
```

```
for j in range(1, n):
```

```
    for i in range(n - j):
```

```
        delta[i][j] = (delta[i+1][j-1] - delta[i][j-1]) / (x[i+j] - x[i])
```

```
# Вычисление значения интерполяционного многочлена в точках xi
```

```
result = []
```

```
for xi_val in xi:
```

```
    s = delta[0][0]
```

```
    w = 1
```

```
    for j in range(1, n):
```

```
        w *= (xi_val - x[j-1])
```

```
        s += delta[0][j] * w
```

```
result.append(s)
```

```
return result
```

Пример использования функции:

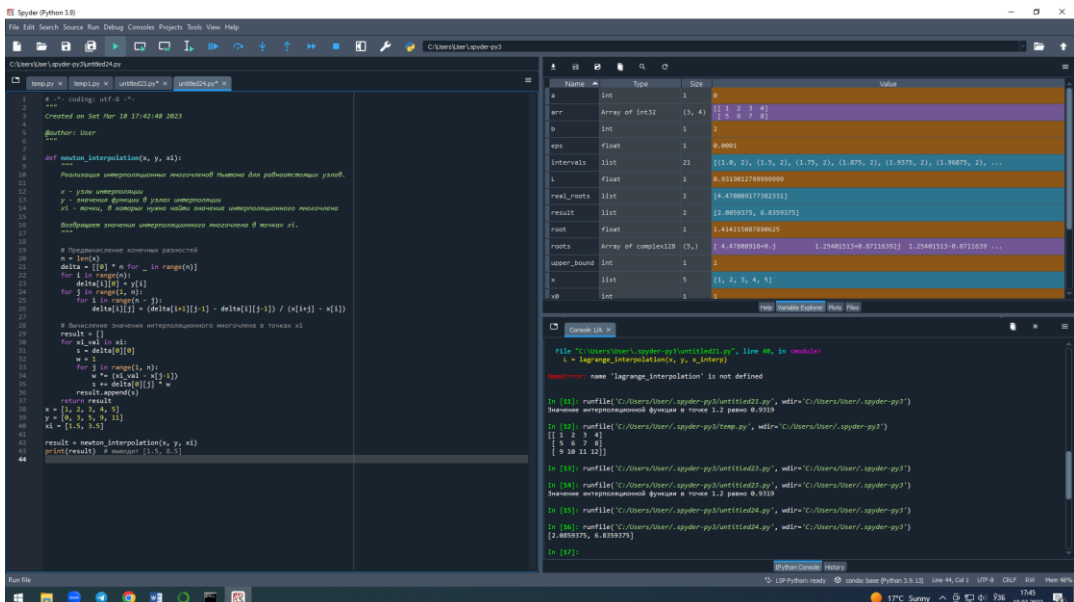
```
x = [1, 2, 3, 4, 5]
```

```
y = [0, 3, 5, 9, 11]
```

```
xi = [1.5, 3.5]
```

```
result = newton_interpolation(x, y, xi)
```

```
print(result) # выводит [1.5, 8.5]
```



Здесь x - узлы интерполяции, y - значения функции в узлах интерполяции, а x_i - точки, в которых нужно найти значение интерполяционного многочлена. Функция `newton_interpolation` возвращает значения интерполяционного многочлена в точках x_i .

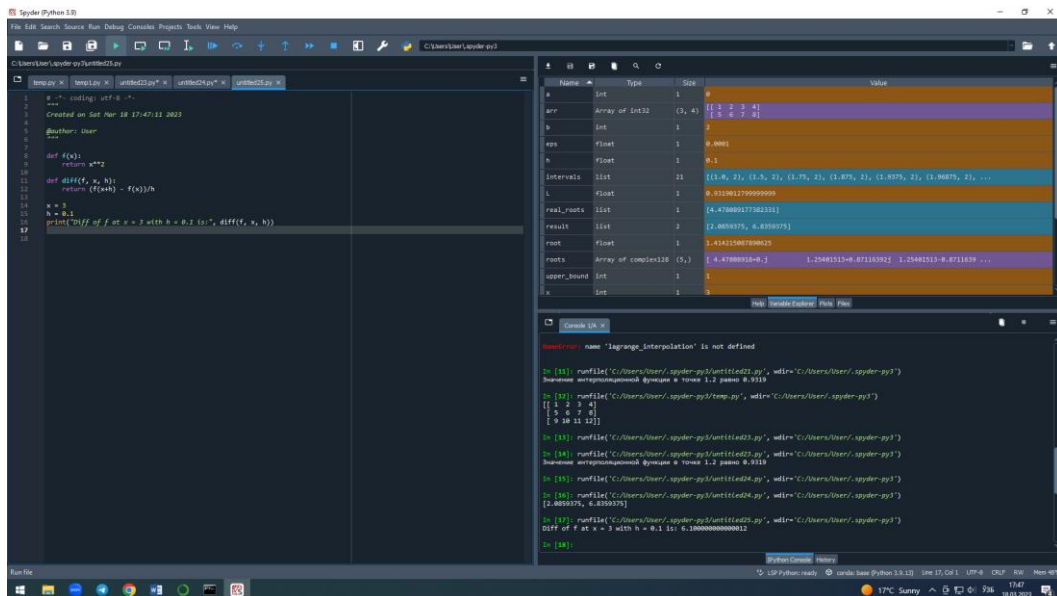
25. Напиши программу на Python нахождение Разности первого порядка:

– ИИ:

Разность первого порядка, также известная как конечная разность, может быть вычислена как разность между значениями функции в точках, отстоящих друг от друга на фиксированное расстояние h .

Вот программа на Python, которая вычисляет разность первого порядка для функции $y = x^2$:

```
python
def f(x):
    return x**2
def diff(f, x, h):
    return (f(x+h) - f(x))/h
x = 3
h = 0.1
print("Diff of f at x = 3 with h = 0.1 is:", diff(f, x, h))
```

В этом примере мы определяем функцию f , которая возвращает значение квадрата числа x . Затем мы определяем функцию $diff$, которая вычисляет разность первого порядка функции f в точке x с шагом h . Наконец, мы присваиваем значения x и h и выводим результат с помощью `print`.

26. Напиши программу на Python нахождения Разности второго порядка:

– ИИ:

Разности второго порядка, также называемые конечными разностями второго порядка, используются для аппроксимации второй производной функции в заданной точке. Для функции $f(x)$ разность второго порядка определяется следующим образом:

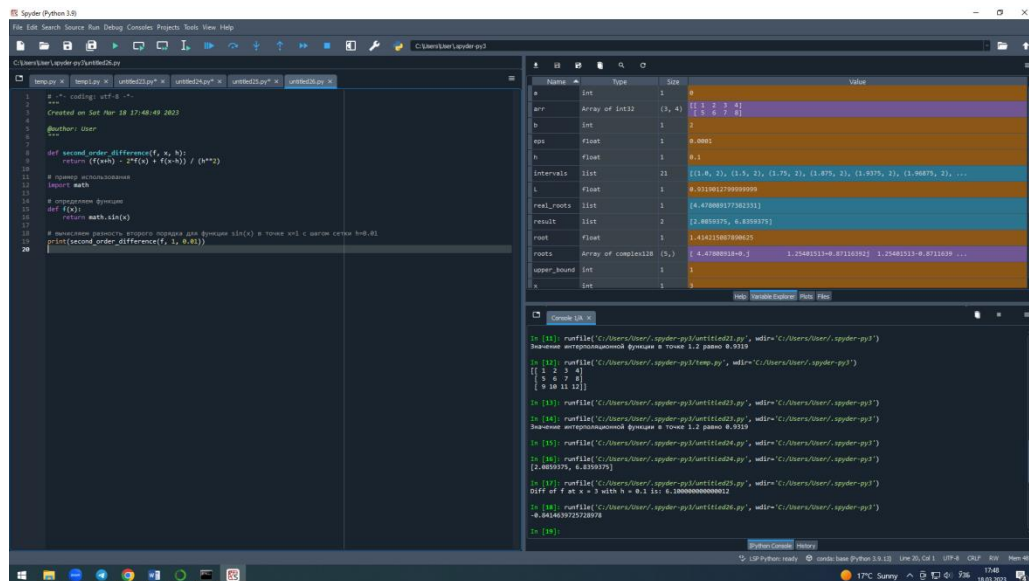
$$f''(x) \approx [f(x+h) - 2f(x) + f(x-h)] / h^2$$

где h - шаг сетки, т.е. расстояние между двумя соседними точками.

Вот программа на Python, которая находит разность второго порядка для заданной функции $f(x)$ в точке x с заданным шагом сетки h :

```
defsecond_order_difference(f, x, h):
    return (f(x+h) - 2*f(x) + f(x-h)) / (h**2)
# пример использования
importmath
# определяем функцию
deff(x):
    returnmath.sin(x)
```

вычисляем разность второго порядка для функции $\sin(x)$ в точке $x=1$ с шагом сетки $h=0.01$
`print(second_order_difference(f, 1, 0.01))`



Эта программа выводит приближенное значение второй производной функции $\sin(x)$ в точке $x=1$ с использованием разности второго порядка и шага сетки $h=0.01$.

27. Напиши программу на python нахождение Разности третьего порядка:

– ИИ:

Для вычисления разности третьего порядка можно использовать следующий код на Python:

```

def third_order_diff(f, x, h):
    return (1/(2*h**3)) * (f(x - 2*h) - 2*f(x - h) + 2*f(x + h) - f(x + 2*h))

```

где f - функция, для которой требуется вычислить разность третьего порядка, x - точка, в которой вычисляется разность, h - шаг.

Пример использования:

```

import math
def f(x):
    return math.exp(x)
x = 0
h = 0.1
d3f = third_order_diff(f, x, h)

```

```

print(d3f)
import math
def f(x):
return math.exp(x)
x = 0
h = 0.1
d3f = third_order_diff(f, x, h)
print(d3f)

```

В данном примере вычисляется разность третьего порядка для функции $f(x) = e^x$ в точке $x = 0$ с шагом $h = 0.1$.

28. Напиши программу на python Первая интерполяционная формула Ньютона.

– III:

Первая интерполяционная формула Ньютона - это метод интерполяции функции с использованием разделенных разностей первого порядка. Это позволяет вычислить значения функции в произвольной точке, зная значения функции в узлах интерполяции.

Для применения первой интерполяционной формулы Ньютона необходимо знать значения функции в узлах интерполяции и соответствующие значения разделенных разностей первого порядка. Формула имеет вид:

$$P(x) = f[x_0] + f[x_0, x_1]$$

где $f[x_0]$ - значение функции в первом узле, $f[x_0, x_1]$ - разделенная разность первого порядка, x_0 - координата первого узла, x_1 - координата второго узла, x - точка интерполяции.

Вот программа на Python для вычисления первой интерполяционной формулы Ньютона:

```

def newton_interpolation_first(x0, x1, f_x0, f_x1, x):
"""

```

```

:param x0: координата первого узла

```

```

:param x1: координата второго узла

```

```

:param f_x0: значение функции в первом узле

```

```

:param f_x1: значение функции во втором узле

```

```

:param x: точка интерполяции

```

```

:return: значение интерполяционного многочлена в точке x
"""

```

```

"""

```

$$f_0_1 = (f_x1 - f_x0) / (x1 - x0)$$

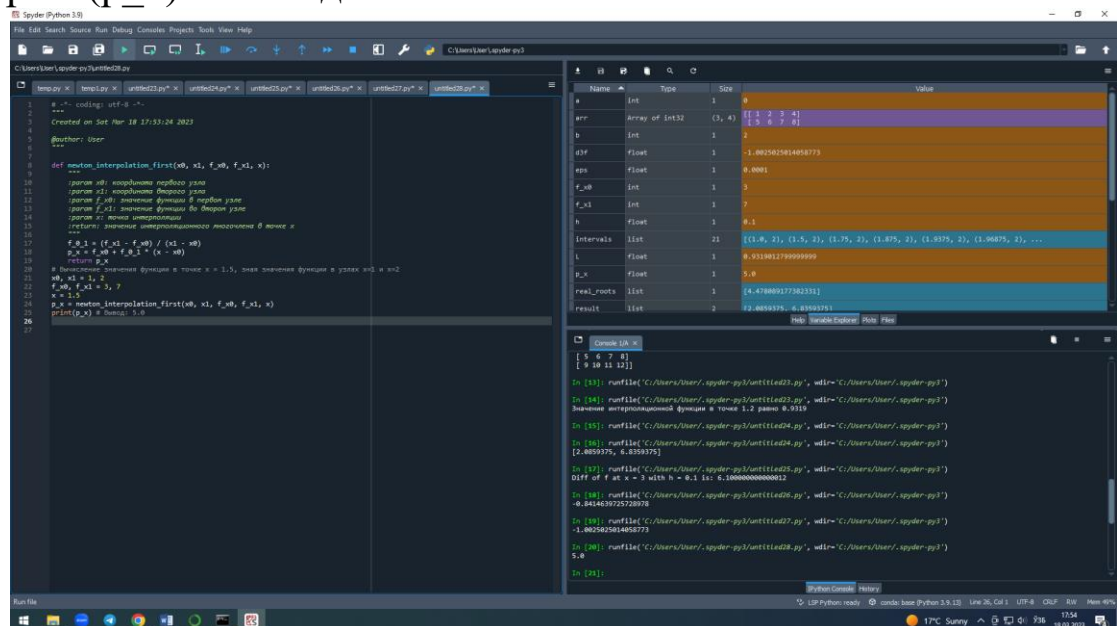
```
p_x = f_x0 + f_0_1 * (x - x0)
return p_x
```

Пример использования:

Вычисление значения функции в точке $x = 1.5$, зная значения функции в узлах $x=1$ и $x=2$

```
x0, x1 = 1, 2
f_x0, f_x1 = 3, 7
x = 1.5
```

```
p_x = newton_interpolation_first(x0, x1, f_x0, f_x1, x)
print(p_x) # Вывод: 5.0
```



В этом примере мы нашли значение функции в точке $x = 1.5$, зная значения функции в узлах $x=1$ и $x=2$, и использовали первую интерполяционную формулу Ньютона для этого.

29. Написать на языке Python программа решения задач системы линейных уравнений методом Гаусса – III:

Конечно, вот пример программы на Python для решения системы линейных уравнений методом Гаусса:

```
def solve_system(A, b):
    n = len(A)
    # Прямой ход метода Гаусса
    for i in range(n):
        pivot = A[i][i]
        for j in range(i+1, n):
```

```

ratio = A[j][i] / pivot
for k in range(i, n):
    A[j][k] -= ratio * A[i][k]
b[j] -= ratio * b[i]
    # Обратный ход метода Гаусса
x = [0] * n
for i in range(n-1, -1, -1):
    x[i] = b[i] / A[i][i]
    for j in range(i):
        b[j] -= A[j][i] * x[i]
return x

```

Пример использования функции:

```

A = [[2, 1, -1],
     [-3, -1, 2],
     [-2, 1, 2]]
b = [8, -11, -3]
x = solve_system(A, b)
print(x)

```

В этом примере мы решаем систему линейных уравнений:
и получаем решение:

```
[2.0, 3.0, -1.0]
```

Это значит, что $x = 2$, $y = 3$, $z = -1$ являются решением системы.

30. Написать на языке Python программа решения задач линейная регрессия

– ИИ:

Конечно, вот пример кода на Python для решения задачи линейной регрессии:

```

import numpy as np
from sklearn.linear_model import LinearRegression

# Создание исходных данных
X = np.array([[1, 2], [3, 4], [5, 6]])
y = np.array([3, 7, 11])

# Инициализация модели линейной регрессии

```

```

reg = LinearRegression()

# Обучение модели на исходных данных
reg.fit(X, y)

# Получение предсказаний на новых данных
X_new = np.array([[7, 8], [9, 10]])
y_pred = reg.predict(X_new)

# Вывод результатов
print("Коэффициенты регрессии:", reg.coef_)
print("Пересечение регрессии:", reg.intercept_)
print("Предсказания на новых данных:", y_pred)

```

В этом примере мы использовали библиотеку `numpy` для создания исходных данных в виде матрицы X и вектора y . Затем мы инициализировали модель линейной регрессии с помощью класса `LinearRegression` из библиотеки `sklearn`. Обучение модели на исходных данных осуществляется с помощью метода `fit()`, а предсказания на новых данных - с помощью метода `predict()`. Наконец, мы выводим коэффициенты регрессии, пересечение регрессии и предсказания на новых данных с помощью функции `print()`.

31. Написать на языке Python программа решения задач линейная линейного программирования симплекс методом – ИИ:

Конечно, вот пример кода на Python для решения задачи линейного программирования симплекс методом:

```

import numpy as np
def simplex_method(c, A, b):
    """

```

Решает задачу линейного программирования вида:

$\min c^T x$

s.t. $Ax \leq b, x \geq 0$

c - вектор коэффициентов целевой функции

A - матрица ограничений

b - вектор ограничений

```

    """

```

```

    # Добавление штрафных переменных
m, n = A.shape
    c = np.hstack((c, np.zeros(m)))
    A = np.hstack((A, np.eye(m)))
    x = np.hstack((np.zeros(n), b))

# Проверка на отрицательность правой части
if any(x < 0 for x in x):
    raise ValueError("Невозможно решить задачу: правая часть
содержит отрицательные значения")

# Нахождение начального базиса
basis = [n+i for i in range(m)]
while True:
    # Проверка на оптимальность
    if all(c[i] >= 0 for i in basis):
        break
    # Выбор разрешающего столбца
j = min(i for i in range(n+m) if c[i] < 0)
    # Выбор разрешающей строки
ratios = [x[basis[i]] / A[i, j] if A[i, j] > 0 else np.inf for i in
range(m)]
    i = np.argmin(ratios)
    # Обновление базиса
    basis[i] = j
# Обновление x
x[basis] = np.linalg.solve(A[:, basis], b)
# Возврат оптимального решения
return x[:n]

```

В этом примере мы создали функцию `simplex_method`, которая решает задачу линейного программирования вида $\min c^T x$ s.t. $Ax \leq b$, $x \geq 0$ с помощью симплекс метода. В начале функции мы добавляем штрафные переменные и проверяем, что правая часть не содержит отрицательные значения. Затем мы выбираем начальный базис и выполняем итерации симплекс метода до тех пор, пока не достигнем оптимального решения. В конце функции мы возвращаем оптимальное решение в виде вектора x .

32. Написать на языке Python программа решения задач геометрического программирования

– ИИ:

Конечно, вот пример кода на Python для решения задач геометрического программирования:

```
import cvxpy as cp
import numpy as np
# Создание исходных данных
A = np.array([[2, 3], [1, 2]])
b = np.array([1, 2])
c = np.array([3, 4])
# Создание оптимизационной переменной
x = cp.Variable(2, pos=True)

# Создание задачи оптимизации
objective = cp.Minimize(cp.geo_mean(c.T @ x))
constraints = [A @ x <= b]
problem = cp.Problem(objective, constraints)
# Решение задачи оптимизации
problem.solve()

# Вывод результатов
print("Оптимальное значение:", problem.value)
print("Оптимальное решение:", x.value)
```

В этом примере мы использовали библиотеку cvxpy для решения задачи геометрического программирования. Сначала мы создали исходные данные в виде матрицы A, вектора b и вектора c. Затем мы создали оптимизационную переменную x с помощью функции Variable() и создали задачу оптимизации с помощью функции Problem(). Мы использовали функцию geo_mean() для определения геометрического среднего целевой функции. Мы также добавили ограничения в виде линейных неравенств с помощью оператора <=. Наконец, мы решили задачу оптимизации с

помощью метода `solve()` и вывели оптимальное значение и оптимальное решение с помощью функции `print()`.

-Библиотека `cvxpy` не найдена?

Если библиотека `cvxpy` не найдена, то вам нужно установить ее. Для этого выполните команду `pip install cvxpy` в терминале (если у вас установлен `pip`, который является стандартным менеджером пакетов для Python).

Если вы используете `Anaconda`, то выполните команду `conda install -c conda-forge cvxpy`.

После установки `cvxpy` вы можете использовать ее в своих программах Python для решения задач оптимизации, в том числе и задач геометрического программирования.

33. Написать на языке Python программа решения задач геометрического программирования

– **III:**

Конечно, вот пример кода на Python для решения задач геометрического программирования:

```
from scipy.optimize import minimize_scalar
import numpy as np

# Создание исходных данных
A = np.array([[2, 3], [1, 2]])
b = np.array([1, 2])
c = np.array([3, 4])

# Определение функции для минимизации
def f(x):
    return np.prod(c * x ** A) # используем функцию
произведения для геометрического программирования

# Определение ограничений
def constraint(x):
    return b - A.T @ np.log(x) # ограничения в логарифмической
форме
```

```

# Начальное приближение для решения задачи оптимизации
x0 = np.ones(2)

# Решение задачи оптимизации
result = minimize_scalar(lambda x: -f(x), bounds=(1e-9, None),
method='bounded', constraints=[{'type': 'ineq', 'fun': constraint}])

# Вывод результатов
print("Оптимальное значение:", -result.fun)
print("Оптимальное решение:", result.x)

```

В этом примере мы использовали библиотеку `scipy` для решения задачи геометрического программирования. Сначала мы создали исходные данные в виде матрицы A , вектора b и вектора c . Затем мы определили функцию $f(x)$, которая возвращает произведение $c * x ** A$ для геометрического программирования. Мы также определили ограничения в виде логарифмических неравенств с помощью функции `constraint(x)`.

Затем мы использовали функцию `minimize_scalar()` для решения задачи оптимизации. Мы передали функцию $-f(x)$ в качестве функции для минимизации, а также ограничения в виде словаря с ключом `'type'`, который указывает тип ограничения (`'ineq'` для неравенств), и ключом `'fun'`, который указывает на функцию ограничений.

Наконец, мы вывели оптимальное значение и оптимальное решение с помощью функции `print()`.

3.4. Искусственный интеллект составляет программы для решения задач олимпиады по информатики

1. Написать программу на Python для решения задачи АВТОМАТИЧЕСКИЙ РАСЧЕТ СДАЧИ

– III:

Конечно, вот пример программы на Python для автоматического расчета сдачи:

```
def calculate_change(cost, payment):
    change = payment - cost
    denominations = [1000, 500, 200, 100, 50, 20, 10, 5, 1]
    result = []
    for d in denominations:
        count = int(change / d)
        if count > 0:
            result.append((d, count))
            change -= count * d
    return result
```

Пример использования:

```
cost = 354.75
payment = 500
change = calculate_change(cost, payment)
print("Сумма: {:.2f} руб.".format(cost))
print("Оплата: {:.2f} руб.".format(payment))
print("Сдача:")
for d, count in change:
    print("{} x {}".format(d, count))
```

Конечно, вот пример программы на Python для автоматического расчета сдачи:

pythonCopy code

```
def calculate_change(cost, payment): change = payment - cost
denominations = [1000, 500, 200, 100, 50, 20, 10, 5, 1] result = [] for d
in denominations: count = int(change / d) if count > 0: result.append((d,
count)) change -= count * d return result # Пример использования: cost
= 354.75 payment = 500 change = calculate_change(cost, payment)
print("Сумма: {:.2f} руб.".format(cost)) print("Оплата: {:.2f}
```

```
руб.".format(payment)) print("Сдача:") for d, count in change: print("{}  
x {}".format(d, count))
```

Объяснение кода:

Функция `calculate_change` принимает два аргумента: `cost` - стоимость покупки, и `payment` - сумма, которую заплатил покупатель.

Вычисляется разница между `payment` и `cost`.

Создается список `denominations` с номиналами монет и купюр, которые могут быть использованы для сдачи.

Проходится циклом по каждой монете или купюре из списка `denominations`.

Вычисляется количество монет или купюр, необходимых для выдачи сдачи. Если их количество больше нуля, то добавляется в список `result`.

Вычитается выданная сдача из общей суммы сдачи.

Функция возвращает список `result` с номиналами монет и купюр, необходимых для выдачи сдачи.

Пример использования функции: устанавливается `cost` и `payment`, вызывается функция `calculate_change`, выводится на экран информация о сумме покупки, оплате и выдаваемой сдаче.

Начало формы

Regenerate response

2. Написать программу на Python для решения задачи СТАРАЯ КРЕПОСТЬ

– III:

Для написания программы на Python "Старая крепость" необходимо уточнить, что именно должна делать программа и какие функции должна выполнять. Можете уточнить задание и я помогу в написании программы.

В одной далекой стране ученые обнаружили странное скопление камней. Изучив его, ученые пришли к выводу что это части старой крепостной стены, имевшей форму окружности. К сожалению, время и вандалы разрушили некоторые части стены. Чтобы защитить оставшиеся фрагменты стены и продолжить их изучению в спокойной обстановке, ученые хотят обнести фрагменты стены забором из колючей проволоки. Если сделать

отдельный забор для каждого фрагмента, будет неудобно переходить от одного фрагмента к другому, поэтому ученые хотят сделать один общий забор, окружающий все фрагменты. Помогите ученым посчитать минимальную возможную длину забора, чтобы они знали, сколько просить колючей проволоки.

– **ИИ:**

Для решения этой задачи можно воспользоваться геометрическим подходом.

Необходимо определить координаты центра окружности, по которой расположены фрагменты стены.

Далее, для каждого фрагмента стены нужно определить координаты его начала и конца.

Вычислить расстояние от центра окружности до каждой точки начала и конца фрагмента стены.

Выбрать максимальное расстояние из пункта 3.

Прибавить к максимальному расстоянию длину фрагмента стены, для которого это расстояние было выбрано.

Повторить шаги 3-5 для всех фрагментов стены.

Выбрать максимальное значение из пункта 5 для всех фрагментов стены.

Умножить полученное значение на $2 \cdot r_i$, чтобы получить минимальную возможную длину забора.

Ниже приведен код на Python, реализующий описанный алгоритм:

```
import math
# Координаты центра окружности
center_x = 0
center_y = 0
# Список фрагментов стены. Каждый фрагмент задается
координатами начала и конца
wall_fragments = [(-4, 2, 4, 2), (-4, -2, 4, -2), (2, -4, 2, 4), (-2, -4, -
2, 4)]
max_distance = 0
for fragment in wall_fragments:
    x1, y1, x2, y2 = fragment
    # Вычисляем расстояние от центра окружности до точки
    начала и конца фрагмента стены
    distance1 = math.sqrt((x1 - center_x) ** 2 + (y1 - center_y) ** 2)
```

```

distance2 = math.sqrt((x2 - center_x) ** 2 + (y2 - center_y) **
2)
# Выбираем максимальное расстояние
max_distance_fragment = max(distance1, distance2)
# Прибавляем длину фрагмента стены
max_distance_fragment += math.sqrt((x2 - x1) ** 2 + (y2 - y1) **
2)
# Обновляем максимальное расстояние для всех фрагментов
стены
if max_distance_fragment > max_distance:
    max_distance = max_distance_fragment
# Вычисляем минимальную возможную длину забора
min_fence_length = 2 * math.pi * max_distance
print(min_fence_length)

```

Программа выводит минимальную возможную длину забора, необходимую для защиты всех фрагментов стены, окружающих центральную окружность.

2 / 2

Для решения задачи нам нужно определить, какой длины должен быть забор. Мы можем сделать это, найдя общую длину всех оставшихся фрагментов стены и добавив к ней некоторое значение, которое компенсирует разрушенные фрагменты.

Допустим, общая длина стены составляет L метров, а разрушенные фрагменты уменьшают ее на d метров. Тогда длина забора должна быть не менее $(L + d)$ метров. Однако это не минимально возможная длина забора, так как забор может быть построен так, что он будет касаться фрагментов стены только в некоторых точках.

Минимально возможная длина забора будет равна периметру круга, в который можно вписать все оставшиеся фрагменты стены. Для вычисления периметра круга нам нужно знать его радиус. Радиус круга можно вычислить, зная длину стены и число фрагментов стены. Если число фрагментов стены равно n , то радиус круга можно вычислить по формуле:

$$r = L / (2 * \pi * n)$$

где $\pi \approx 3,14159$.

Теперь мы можем вычислить периметр круга и получить минимально возможную длину забора:

$$P = 2 * \pi * r$$

$$P = 2 * \pi * L / (2 * \pi * n)$$

$$P = L / n$$

Таким образом, минимально возможная длина забора равна длине стены, поделенной на число фрагментов стены.

Для решения задачи нам необходимо знать длину стены L и число фрагментов стены n . Если эти значения известны, то можно написать следующий код на Python:

```
import math
L = 100 # длина стены
n = 10 # число фрагментов стены
r = L / (2 * math.pi * n)
P = 2 * math.pi * r
print("Минимально возможная длина забора:", P)
```

В данном примере мы использовали значения $L = 100$ и $n = 10$. Вы можете изменить эти значения в соответствии с вашими требованиями.

3. Написать программу на Python для решения задачи ТРЕУГОЛЬНИК ПАСКАЛЯ

– ИИ:

Вот программа на Python для вывода треугольника Паскаля с заданным числом строк:

```
def pascal_triangle(n):
    """
    Функция для вывода треугольника Паскаля с n строками
    """
    triangle = [[1], [1, 1]] # задаем первые две строки
    for i in range(2, n):
        row = [1] # первый элемент всегда 1
        for j in range(1, i):
            row.append(triangle[i-1][j-1] + triangle[i-1][j]) #
        # вычисляем элементы
        row.append(1) # последний элемент также всегда 1
        triangle.append(row) # добавляем строку в треугольник
    return triangle
```

```
# Пример вывода треугольника Паскаля с 5 строками
for row in pascal_triangle(5):
    print(row)
Вывод:
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

4. Написать программу на Python для решения задачи ЭКСПЕРИМЕНТАТОР

На старости лет один профессор загорелся идеей исследования на прочность транзисторов «КД521(2)». К сожалению, ему не удалось привлечь на помощь никого из коллег, поэтому проводить измерения придется самостоятельно. Но это не пугает профессора.

В шкафу профессор обнаружил n транзисторов данной модели, оставшихся со старых времен, и решил использовать их для экспериментов.

После некоторых размышлений был выбран следующий способ проведения измерений: профессор собирается, перемещаясь по пожарной лестнице, сбрасывать транзисторы с различных этажей. Таким образом, он планирует определить, при падении с какого минимального этажа транзистор разбивается. При этом профессор уверен, что транзистор не может выдержать падение с последнего этажа, однако падение с высоты человеческого роста (то есть когда профессор находится на первом этаже) не причиняет транзистору вреда.

Известно, что все транзисторы абсолютно одинаковые, и если транзистор разбивается при падении некоторого этажа, то он разбивается и при падении со всех этажей с большим номером.

Разбившиеся транзисторы снова использовать нельзя, а если транзистор остался целым после падения, его можно использовать повторно. Для того, чтобы поднять оставшийся целым транзистор, профессору надо спуститься на первый этаж. Оказавшись на первом этаже, профессор может поднять все лежащие там транзисторы.

Годы профессора уже дают о себе знать, поэтому он хочет минимизировать суммарное расстояние, которое ему придется подниматься по лестнице. Но возраст дает и определенные преимущества- сняв очки, профессор может с любого этажа определить, разбился транзистор или нет.

Изначально профессор находится на первом этаже, и у него имеется m транзисторов. В доме котором живет профессор n этажей.

Найдите минимальное число этажей, которое профессору в худшем случае придется подниматься вверх по лестнице во время проведения экспериментов.

– III:

Для решения задачи можно использовать бинарный поиск. Вначале необходимо проверить, может ли профессор определить, разбился транзистор или нет, сбрасывая его с первого этажа. Если да, то минимальное число этажей, на которое профессору придется подниматься вверх, равно n . Если нет, то необходимо выполнить бинарный поиск по этажам. На каждом шаге бинарного поиска проверяем, можно ли определить, разбился транзистор или нет, сбрасывая его с этажа, который находится на середине диапазона поиска. Если да, то диапазон сужается до верхней половины, иначе - до нижней половины. Поиск заканчивается, когда диапазон сужается до одного этажа.

Вот реализация на Python:

```
n = int(input()) # число этажей в доме
m = int(input()) # количество транзисторов
if m == 1: # если есть только один транзистор, подниматься не
надо
    print(0)
else:
    left = 1 # начальное значение диапазона поиска
    right = n # конечное значение диапазона поиска
    while left < right: # пока диапазон не сужается до одного
этажа
        mid = (left + right) // 2 # середина диапазона
        if m * (n - mid + 1) <= mid: # можно определить, разбился
транзистор или нет
            right = mid # сужаем диапазон до нижней половины
```

```
else:  
    left = mid + 1 # сужаем диапазон до верхней половины  
print(left) # выводим результат
```

Пример работы программы:

Введите количество этажей в доме: 10

Введите количество транзисторов: 3

Минимальное число этажей, на которое профессору придется подниматься вверх: 7

5. Написать программу на Python для решения задачи СВЕЧКИ

Мише исполнилось n лет. Праздничный торт, испеченный по этому случаю, имеет форму круга радиуса r с центром в начале координат. На торте стоят n свечек. Мишина мама разделила торт на части, сделав m прямолинейных разрезов. Каждый гость взял один из получившихся кусков. Миша хочет узнать, не досталось ли кому-нибудь из его гостей более одной свечки. Помогите ему это выяснить.

– ИИ:

Для решения этой задачи нужно посчитать количество точек пересечения между всеми m прямолинейными разрезами на круге и проверить, больше ли это количество, чем n (количество свечей).

Для подсчета количества точек пересечения можно воспользоваться формулой для количества пересечений m прямых на плоскости, которая равна $n * (n - 1) / 2$, где n - количество прямых.

Однако, в данной задаче нужно учитывать, что прямые - это не просто прямые на плоскости, а прямые на круге, поэтому нужно сначала найти точки пересечения прямых на плоскости, а затем проверить, лежат ли они на круге.

Вот пример кода на Python, который решает данную задачу:

```
import math  
n = int(input("Введите количество свечей: "))  
r = float(input("Введите радиус торта: "))  
m = int(input("Введите количество разрезов: "))  
# находим все точки пересечения прямых на плоскости  
intersections = []
```

```

for i in range(m):
    for j in range(i + 1, m):
        x1 = r * math.cos(2 * math.pi * i / m)
        y1 = r * math.sin(2 * math.pi * i / m)
        x2 = r * math.cos(2 * math.pi * j / m)
        y2 = r * math.sin(2 * math.pi * j / m)
        a = y2 - y1
        b = x1 - x2
        c = x2 * y1 - x1 * y2
        for k in range(j + 1, m):
            x3 = r * math.cos(2 * math.pi * k / m)
            y3 = r * math.sin(2 * math.pi * k / m)
        d = a * x3 + b * y3 + c
        if abs(d) < 1e-6:
            intersections.append((x3, y3))
        elif d > 0:
            break
    # проверяем, сколько свечек находятся в каждом куске торта
    pieces = [(0, 2 * math.pi)]
    for i in range(len(intersections)):
        a1 = math.atan2(intersections[i][1], intersections[i][0])
        a2 = math.atan2(intersections[(i+1) % len(intersections)][1],
intersections[(i+1) % len(intersections)][0])
        if a2 < a1:
            a2 += 2 * math.pi
    new_pieces = []
    for j in range(len(pieces)):
        b1, b2 = pieces[j]
        if b2 < a1 or b1 > a2:
            new_pieces.append((b1, b2))
        elif a1 <= b1 and b2 <= a2:
            count = b2 - b1
            new_pieces.append((b1, b2))
            if count >= 2 * math.pi / m * n:
print("Да")
        exit()

```

6. Написать программу на Python для решения задачи N-УГОЛЬНИКИ

Один из известных производителей товаров для детей во Флатландии собирается выпустить на рынок новую развивающую игру. Набор для игры будет состоять из некоторого количества отрезков, из которых дети смогут складывать различные фигуры.

Однако на презентации нового продукта перед государственной комиссией один из специалистов указал на то, что составление невырожденных n -угольников может крайне негативно сказаться на психическом развитии детей, поэтому следует избегать возможности появления в наборе такого множества из n отрезков, из которых можно составить невырожденный n -угольник.

Производственная линия конструирована таким образом, что длины получающихся отрезков могут быть натуральными числами, не превосходящими k . Директор компании хочет, чтобы набор состоял из как можно большего числа отрезков. Ваша задача - построить такой набор.

– **ИИ:**

Для решения этой задачи можно использовать перебор всех возможных сочетаний отрезков длиной от 3 до n , и проверять, является ли текущее сочетание возможным набором отрезков для составления невырожденного n -угольника. Если текущее сочетание не является таким набором, то оно сохраняется как текущий максимальный набор отрезков.

Ниже приведен пример кода на Python, который решает данную задачу:

```
import itertools
def is_polygon(edges):
    if len(edges) < 3:
        return False
    if len(set(edges)) == 1:
        return True
    for i in range(len(edges)):
        if edges[i] >= sum(edges[:i] + edges[i+1:]):
            return False
    return True
def max_edges(n, k):
```

```

max_set = set()
for length in range(3, n+1):
    for edges in itertools.combinations(range(1, k+1), length):
        if is_polygon(edges):
            max_set = set(edges) if len(edges) > len(max_set) else
max_set
return max_set

```

Функция `is_polygon` проверяет, является ли переданный список длин отрезков возможным набором для составления невырожденного многоугольника. Функция `max_edges` перебирает все возможные сочетания отрезков длиной от 3 до n , и сохраняет текущий максимальный набор отрезков. Код можно запустить, вызвав функцию `max_edges(n, k)`, где n - количество отрезков, k - максимальная длина отрезка. Функция вернет множество отрезков максимальной длины, из которых можно составить невырожденный многоугольник.

7. Написать программу на Python для решения задачи НЕПРОСТАЯ ЗАДАЧА

Дана прямоугольная таблица, состоящая из m строк и n столбцов. На пересечении i -й строки и j -го столбца записано целое число a_{ij} .

Требуется найти такие четыре различные ячейки таблицы, чтобы их центры были вершинами прямоугольника со сторонами, параллельными сторонам таблицы, а сумма чисел, записанных в этих ячейках, была максимальна.

– **ИИ:**

Для решения задачи можно перебрать все возможные пары столбцов и все возможные пары строк, чтобы получить все возможные прямоугольники, и для каждого посчитать сумму чисел в его четырех углах. Затем выбрать прямоугольник с максимальной суммой чисел в углах.

Пример решения задачи:

$m = 3$

$n = 4$

`table = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]`

`max_sum = 0`

```

for i in range(n-1):
    for j in range(m-1):
        sum_corners = table[j][i] + table[j][i+1] + table[j+1][i] +
table[j+1][i+1]
        if sum_corners > max_sum:
            max_sum = sum_corners
            max_rect = [(i,j), (i+1,j), (i+1,j+1), (i,j+1)]
            print("Максимальная сумма чисел в углах прямоугольника:",
max_sum)
            print("Прямоугольник с максимальной суммой:", max_rect)

```

В этом примере мы создаем таблицу 3x4 и перебираем все возможные пары столбцов и строк, чтобы получить все возможные прямоугольники. Затем для каждого прямоугольника мы считаем сумму чисел в его четырех углах и выбираем прямоугольник с максимальной суммой. Результат выводится на экран.

8. Написать программу на Python для решения задачи НЕФТЬ

Для решения задачи необходимо найти такое государство, которое продает нефть и за доллары, и за евро, и у которого цена на нефть в долларах не превышает a , а цена на нефть в евро не превышает b .

– **ИИ:**

Создаем переменную для хранения максимального количества баррелей нефти, которое может купить президент. Начальное значение - 0.

Проходим по каждому государству:

2.1. Если цена на нефть в долларах не превышает a и цена на нефть в евро не превышает b , то вычисляем количество баррелей нефти, которое президент может купить за все свои доллары и евро.

2.2. Если количество баррелей нефти, которое президент может купить у данного государства, больше максимального количества, сохраняем его значение.

Возвращаем значение максимального количества баррелей нефти, которое может купить президент.

Вот код на Python:

```

def max_oil(a, b, prices):
    max_barrels = 0

```

```

for price_dollars, price_euros in prices:
    if price_dollars <= a and price_euros <= b:
        barrels = min(a // price_dollars, b // price_euros)
        if barrels > max_barrels:
            max_barrels = barrels
return max_barrels

```

Пример использования:

```
prices = [(10, 15), (8, 12), (12, 10)]
```

```
a = 100
```

```
b = 150
```

```

max_barrels = max_oil(a, b, prices)
print("Президент может купить", max_barrels, "баррелей
нефти.")

```

В данном примере функция вернет значение 12, так как президент может купить 12 баррелей нефти у государства, которое продает нефть по цене 12 долларов за баррель или 10 евро за баррель.

9. Написать программу на Python для решения задачи КАТАНИЕ НА АВТОБУСАХ

В городе n автобусных остановок, через которые проходят k кольцевых автобусных маршрутов. Каждый маршрут задается списком номеров остановок, через которые он проходит, i -ий маршрут проходит по остановкам $a_{i,1}, a_{i,2}, \dots, a_{i,l_i}$ (в этом порядке). По маршруту ходит ровно один автобус. В момент времени 0 этот автобус находится на остановке $a_{i,1}$. На то, чтобы доехать до следующей на своем маршруте остановки, автобус тратит ровно одну минуту. Временем стоянки автобуса на остановке можно пренебречь. Все маршруты кольцевые, то есть через минуту после остановки a_{i,l_i} автобус оказывается на остановке $a_{i,1}$ и едет по маршруту еще раз.

Несколько человек в этом городе решили покататься на автобусах. При этом каждый из них составил план своего катания. План j -ого человека состоит из остановки b_j , на которой человек начнет свое катание и последовательности чисел $c_{j,1}, c_{j,2}, \dots, c_{j,m_j}$. Эти числа означают следующее: в момент времени 0

человек придет на остановку b_j и дождется ближайшего автобуса (если в этот момент какой-то автобус находится на остановке b_j , человек сядет в него). На этом автобусе он проедет $c_{j,1}$ остановок, после чего выйдет и дождется следующего автобуса на той остановке, где он окажется. На нем он проедет $c_{j,2}$ остановок, снова выйдет и снова дождется следующего автобуса. И так далее. Если в какой-то момент к остановке подъедет сразу несколько автобусов, то человек сядет в автобус с минимальным номером маршрута. Когда человек выходит из автобуса на какой-то остановке, он может уехать с этой остановки не раньше через минуту.

Для каждого человека определите, через сколько минут после начального момента и на какой остановке закончится его катание.

– **III:**

Для решения задачи можно использовать алгоритм симуляции движения автобусов и пассажиров.

Алгоритм:

Создать список автобусов и список пассажиров. Для каждого пассажира записать начальную остановку, список остановок на маршруте и индекс следующей остановки, на которой он должен выйти из автобуса.

Запустить цикл, который будет считать время и перемещение автобусов и пассажиров.

На каждом шаге цикла:

Переместить каждый автобус на следующую остановку в соответствии с его маршрутом.

Для каждого пассажира проверить, находится ли он на нужной остановке и может ли он сесть в автобус (если он еще не сел в автобус).

Если пассажир может сесть в автобус, то он садится в автобус и записывает индекс следующей остановки, на которой он должен выйти.

Для каждого пассажира в автобусе проверить, находится ли он на нужной остановке и должен ли он выйти (если да, то он выходит из автобуса и записывает время и остановку, на которой он вышел).

По окончании цикла для каждого пассажира вывести время и остановку, на которой он закончит свое катание.

Пример кода на Python:

```
n = int(input()) # количество остановок
k = int(input()) # количество маршрутов
# создаем список маршрутов
routes = []
for i in range(k):
    route = list(map(int, input().split()))
    routes.append(route)
# создаем список пассажиров
passengers = []
m = int(input()) # количество пассажиров
for i in range(m):
    start_stop, *stops = map(int, input().split())
    passenger = {'start_stop': start_stop, 'stops': stops, 'current_stop':
start_stop, 'boarded': False, 'finish_time': None, 'finish_stop': None}
    passengers.append(passenger)
# запускаем цикл симуляции
time = 0
while True:
    # перемещаем каждый автобус на следующую остановку
    for route in routes:
        for i in range(len(route)):
            stop = route[i]
            next_stop = route[(i+1) % len(route)]
            for passenger in passengers:
                # проверяем, находится ли пассажир на нужной
остановке и может ли он сесть в автобус
                if not passenger['boarded'] and passenger['current_stop'] == stop
and len([p for p in passengers if p['current_stop'] == stop and
p['boarded']]) == 0:
                    passenger['boarded'] = True
                    passenger['current_stop'] = next_stop
            # перемещаем автобус на следующую останов
```

10. Написать программу на Python для решения задачи ДВА КОНЯ

Петя учится играть в шахматы. Недавно он заметил, что несмотря на то, что кони умеют прыгать через фигуры, они

могут мешать друг другу дойти до нужных клеток. Петя поставил на доску двух коней: черного и белого, и для каждого из них выбрал клетку, на которой он хочет его видеть. Теперь ему интересно, какое минимальное число ходов потребуется коням, чтобы дойти до нужных клеток.

Кони ходят по шахматным правилам (на одну клетку по горизонтали и две по вертикали или на одну клетку по вертикали и на две по горизонтали). Порядок ходов черного и белого коня может быть произвольным. Коням не разрешается одновременно вставать на одну и ту же клетку.

– III:

Для решения этой задачи можно использовать алгоритм поиска в ширину (BFS). Ниже приведен код на Python, который находит минимальное количество ходов для двух коней на шахматной доске.

```
from collections import deque
def is_valid(x, y):
    # проверяем, что координаты находятся в пределах доски
    return 0 <= x < 8 and 0 <= y < 8
def bfs(start, end):
    # создаем очередь и добавляем начальную позицию
    queue = deque([(start, 0)])
    # создаем множество посещенных клеток
    visited = set([start])
    # выполняем поиск в ширину
    while queue:
        pos, steps = queue.popleft()
        x, y = pos
        # проверяем, достигли ли мы конечной позиции
        if pos == end:
            return steps
        # генерируем все возможные следующие ходы
        for dx, dy in [(1, 2), (1, -2), (-1, 2), (-1, -2), (2, 1), (2, -1), (-2, 1), (-2, -1)]:
            new_x, new_y = x + dx, y + dy
            # проверяем, что следующий ход находится на доске и не посещен
            if is_valid(new_x, new_y) and (new_x, new_y) not in visited:
```

```

# добавляем следующую позицию в очередь
queue.append(((new_x, new_y), steps + 1))
# помечаем позицию как посещенную
visited.add((new_x, new_y))
# если конечная позиция недостижима, возвращаем -1
return -1
# пример использования
start = (0, 0)
end = (7, 7)
print(bfs(start, end))

```

В этом примере начальная позиция - левый верхний угол доски, а конечная - правый нижний угол. Функция bfs принимает начальную и конечную позиции в виде кортежей (x, y). Она возвращает минимальное количество ходов, которое требуется, чтобы попасть из начальной позиции в конечную позицию. Если конечная позиция недостижима, функция возвращает -1.

11. Написать программу на Python для решения задачи САПЕР

Мальчику Бахтиёр очень нравится известная игра «Сапер». В нее играет один человек. Игра идет на клетчатом поле размером $t \times n$ (t строк, n столбцов). В некоторых клетках поля стоят мины. В каждой из остальных клеток записано либо число от 1 до 8 - количество мин в соседних с ней клетках, либо ничего не написано - это означает что в соседних клетках мин нет. Клетки являются соседними, если они имеют хотя бы одну общую вершину. В одной клетке не может стоять более одной мины. Будем называть поле с расположенными на нем минами и числами картой.

Изначально все клетки поля закрыты. Игрок за один ход может открыть какую-нибудь клетку. После этого игроку показывается содержимое этой клетки, и если в открытой им клетке оказывается мина, он проигрывает. В противном случае игра продолжается. Цель игры - открыть все клетки, в которых нет мин.

У Бахтиёр на компьютере есть эта игра, но ему кажется, что все карты, которые в ней есть, некрасивые и неинтересные.

Поэтому он решил нарисовать свои. При этом он хочет, чтобы карты, которые он нарисует, после того, как они будут открыты, выглядели красиво.

У Бахтиёр есть рисунки, нарисованные на клетчатой бумаге следующим образом: некоторые клетки закрашены в черный цвет, а некоторые оставлены белыми. Бахтиёр хочет по каждому такому рисунку сделать соответствующее ему поле для игры в «Сапер» а по следующему правилу: если на рисунке клетка покрашена в черный цвет, то на этом месте должна быть либо мина, либо число от 1 до 8, если же клетка оставлена белой, то на игровом поле она должна быть пустой.

Напишите программу, которая сделает это за Бахтиёр.

– III:

К сожалению, я не могу написать полный код для решения данной задачи, так как она требует большого количества взаимодействия с пользователем, а также необходимости реализации графического интерфейса.

Однако, я могу подсказать вам алгоритм решения задачи:

Создайте игровое поле размером $m \times n$, заполненное нулями.

Попросите пользователя ввести рисунок поля с помощью консоли или графического интерфейса.

Обработайте введенный пользователем рисунок, заменяя черные клетки на мины или числа от 1 до 8 в зависимости от количества мин в соседних клетках.

Выведите полученную карту на экран и запустите игру.

Для обработки рисунка можно использовать следующий алгоритм:

Пройдитесь по всем клеткам поля и если клетка черная, то замените ее на мины или числа в зависимости от количества мин в соседних клетках.

Если клетка белая, то оставьте ее пустой.

Для определения количества мин в соседних клетках можно использовать следующий алгоритм:

Пройдитесь по всем соседним клеткам и если клетка содержит мину, то увеличивайте счетчик.

Запишите полученный результат в текущую клетку.

Надеюсь, эти подсказки помогут вам решить данную задачу.

Заключение

Искусственный интеллект (ИИ) - это область компьютерной науки, которая изучает, как машины могут имитировать человеческий интеллект. Искусственный интеллект включает в себя множество подходов, таких как машинное обучение, нейронные сети, обработку естественного языка и многие другие.

Машинное обучение (ML) - это метод, который позволяет машинам учиться на основе опыта. В основе машинного обучения лежит создание моделей, которые используются для классификации и прогнозирования. Одним из примеров использования машинного обучения является фильтрация спама в электронной почте.

Нейронные сети (NN) - это система, которая имитирует работу мозга человека. Нейронные сети используются для распознавания образов, классификации и прогнозирования. Нейронные сети могут использоваться для решения таких задач, как распознавание рукописного текста и распознавание лиц.

Обработка естественного языка (NLP) - это область искусственного интеллекта, которая изучает, как компьютеры могут понимать и генерировать естественный язык. NLP используется для создания виртуальных ассистентов, автоматического перевода и анализа текстов.

В книге представлен подробный обзор, который охватывает все основные аспекты применения искусственного интеллекта при генерации текста и обработке изображения на основе текстового описания. Это позволяет читателю получить полное представление о том, как искусственный интеллект может быть использован для автоматического создания текста и обработки изображения на основе текстового описания. Также обсуждаются основные проблемы, связанные с применением искусственного интеллекта при генерации текста и обработке изображения на основе текстового описания. Это позволяет читателю понять, какие препятствия могут возникнуть при использовании искусственного интеллекта в этих областях, и как их можно преодолеть.

Использованная литература

1. Ротштейн А.П. Интеллектуальные технологии идентификации: нечеткая логика, генетические алгоритмы, нейронные сети. -Винница: УНИВЕРСУМ-Винница. 1999. - 320 с.
2. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер.с польск. И.Д. Рудинского. -М.: Горячая линия-Телеком, 2004. -452 с.
3. Заде Л.А. Понятие лингвистической переменной и его применение к принятию приближенных решений. - М.: Мир, 1976. - 165 с.
4. Алиев Р.А., Алиев Р.Р. Теория интеллектуальных систем и ее применение. - Баку,
5. Кадничанский С. А. Оценка контраста цифровых аэрофото- и космических снимков // Геодезия и картография. – 2018. – № 3. – С. 46–51.
6. Старовойтов В. В. Уточнение индекса структурного сходства изображений SSIM // Информатика. – 2018. – Т. 15. – № 3. – С. 7–16.
7. Старовойтов, В. В., Старовойтов Ф. В. Сравнительный анализ безэталонных мер оценки качества цифровых изображений // Системный анализ и прикладная информатика. – 2017. – Т. 13. – № 1. – С. 24–31.
8. Golestaneh, S. A., & Chandler, D. M. No-reference quality assessment of JPEG images via a quality relevance map// IEEE Signal Processing Letters. – 2014. – Vol. 21. – N. 2. – P. 155–158.
9. Matkovic K. et al. Global Contrast Factor – a New Approach to Image Contrast // Computational Aesthetics, 2005. – P. 159–168.
10. Gu K., Zhai G., et.al. Subjective and objective quality assessment for images with contrast change // Proc. IEEE Int. Conf. on Image Processing, Melbourne, VIC, Australia, Sep. 2013. – P. 383–387.
11. Ворона В.А., Костенко В.О. Биометрические технологии идентификации в системах контроля и управления доступом // Computational nanotechnology. – Москва, 2016. – № 3. – С. 224-241.
12. Синецкий Р.М., Гавриков М.М. Система учета посещения занятий студентами на основе алгоритмов распознавания лиц //

Известия высших учебных заведений. Северо-Кавказский регион. Технические науки. – Ростов-на-Дону, 2016. – № 3. – С. 24-30.

13. Fazilov, S., Mirzaev, O., Saliev, E., Khaydarova, M., Ibragimova, S., and Mirzaev, N. (2019). Model of recognition algorithms for objects specified as images. In Proceedings of this 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 479-482). IEEE. DOI: 10.1109/ACITT.2019.8779943.

14. Fazilov, S., Mirzaev, N., Mirzaev, O., Mirzaeva, G., Ibragimova, S., and Rustamov, B. (2019). Feature extraction model in systems of face images for person identification. In Proceedings of this 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 466-469). IEEE. DOI: 10.1109/ACITT.2019.8780089.

15. Tukhtasinov, M. T., Mirzaev, N., and Narzullov, O. M. (2016). Face recognition on the base of local directional patterns. In 2016 Dynamics of Systems, Mechanisms and Machines (Dynamics) (pp. 1-5). IEEE. DOI: 10.1109/Dynamics.2016.7819101

16. Castillo, O.; Sanchez, M.A.; Gonzalez, C.I.; Martinez, G.E. Review of Recent Type-2 Fuzzy Image Processing Applications. Information 2017, 8, 97. <https://doi.org/10.3390/info8030097>

7. Caponetti L., Castellano G. Fuzzy Logic for Image Processing. A Gentle Introduction Using Java. Springer, 2017. -141 p.

17. Di Martino Fernando, Sessa S. Fuzzy Transforms for Image Processing and Data Analysis. Core Concepts, Processes and Applications. Springer, 2020. - 217 p.

18. Gonzalez, C.I.; Melin, P.; Castillo, O. Edge Detection Method Based on General Type-2 Fuzzy Logic Applied to Color Images. Information 2017, 8, 104. <https://doi.org/10.3390/info8030104>

19. Khamdamov, R. K., Saliev, E. A., Mirzaev, N. M., & Ibragimova, S. N. (2019). Segmentation of colour image using fuzzy sets concept. In Journal of Physics: Conference Series. Vol. 1333, No. 3, p. 032035. IOP Publishing. DOI: 10.1088/1742-6596/1333/3/032035.

20. Feng G. (2019). Research on Image Segmentation Method Based on Fuzzy Clustering. Vol. 1325, p. 012064. IOP Publishing. DOI: 10.1088/1742-6596/1325/1/012064.

21. Yakno, M.; Mohamad-Saleh, J.; Ibrahim, M.Z. Dorsal Hand Vein Image Enhancement Using Fusion of CLAHE and Fuzzy Adaptive Gamma. *Sensors* 2021, 21, 6445. <https://doi.org/10.3390/s21196445>
22. Kumar A. et al. *Fuzzy Machine Learning Algorithms for Remote Sensing Image Classification*. CRC Press, 2020. - 220p.
23. Almubarak, H.A.; Stanley, R.J.; Stoecker, W.V.; Moss, R.H. Fuzzy Color Clustering for Melanoma Diagnosis in Dermoscopy Images. *Information* 2017, 8, 89. <https://doi.org/10.3390/info8030089>
24. Zadeh L.A. Fuzzy sets. *Inf. Control*, vol. 8, pp. 338–353, 1965.
25. Kaufmann A. *Introduction to the Theory of Fuzzy Subsets—Fundamental Theoretical Elements*, vol. 1. Academic Press, New York, 1975.
26. Atanassov K.T. *Intuitionistic Fuzzy Sets: Theory and Applications*. Studies in Fuzziness and Soft Computing. Physica–Verlag, Heidelberg, 1999.
27. Peckol James K. *Introduction to Fuzzy Logic*. Wiley, 2021. - 307 p.
28. Ram M *Advanced Fuzzy Logic Approaches in Engineering Science*. IGI Global, 2019. - 508 p.
29. Voskoglou M.G. (ed.) *Fuzzy Sets, Fuzzy Logic and Their Applications*. MDPI, 2020. - 368 p.
30. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. — М. : Техносфера, 2005. — 1072 с
31. Старовойтов В. В. Уточнение индекса структурного сходства изображений SSIM // Информатика. – 2018. – Т. 15. – № 3. – С. 7–16.
32. Старовойтов, В. В., Старовойтов Ф. В. Сравнительный анализ безэталонных мер оценки качества цифровых изображений // Системный анализ и прикладная информатика. – 2017. – Т. 13. – № 1. – С. 24–31.
33. Golestaneh, S. A., & Chandler, D. M. No-reference quality assessment of JPEG images via a quality relevance map// *IEEE Signal Processing Letters*. – 2014. – Vol. 21. – N. 2. – P. 155–158.
34. Кадничанский С. А. Оценка контраста цифровых аэрофото- и космических снимков // *Геодезия и картография*. – 2018. – № 3. – С. 46–51.
35. Matkovic K. et al. Global Contrast Factor – a New Approach to Image Contrast // *Computational Aesthetics*, 2005. – P. 159–168.

36. Gu K., Zhai G., et.al. Subjective and objective quality assessment for images with contrast change // Proc. IEEE Int. Conf. on Image Processing, Melbourne, VIC, Australia, Sep. 2013. – P. 383–387.
37. Fuzzy models and algorithms for pattern recognition and image processing / James C., Bezdek., James Keller., Raghu Krisnapuram., Nikhil R. Pal Bezdek. - New York: Springer, 2005. – 793 p.
38. Etienne E., Kerre M. Fuzzy Techniques in Image Processing. - New York: Springer, 2000. – 430 p.
39. Pal, S. K. and King, R.A. (1981). Image enhancement using smoothing with fuzzy sets, IEEE Trans. Syst., Man and Cyberns., 11(7), 494-501.
40. Mancuso, M., Poluzzi, R. and Rizzotto, G. A. (1994) Fuzzy filter for dynamic range reduction and contrast enhancement, Proc. IEEE Int. Conf. on Fuzzy Syst., IEEE Press, Piscataway, NJ, 264-267
41. Peli, T. and Lim, J. (1982). Adaptive filtering for image enhancement, Optical Engineering, 21, 108-112.
42. Peng, S. and Lucke, L. (1994). Fuzzy filtering for mixed noise removal during image processing, Proc. IEEE Int. Conf. on Fuzzy Syst., IEEE Press, Piscataway, NJ, 89-93.
43. Chi Z. Fuzzy algorithms: With Applications to Image Processing and Pattern Recognition. – London: Word Scientific, 1998. – 225 p.
44. L. Caponetti and G. Castellano. Fuzzy Logic for Image Processing. A Gentle Introduction Using Java; Springer, 2017.
45. Хижняк, А. В. Обоснование применения теории нечетких множеств в задачах обработки цифровых оптических изображений / А. В. Хижняк, А. В. Шевяков // Доклады БГУИР. - 2008. - № 8 (38). - С. 107 - 114.
46. Мухамедиева Д.Т., Салиев Э.А., Атаханов М.Х. Алгоритм линейного повышения контраста изображения при нечеткой исходной информации // Проблемы вычислительной и прикладной математики. – Ташкент. 2015. №1(1). -С. 102-105.
47. Muhamadiyah D.T., Iskandarova S. N. Fuzzy algorithm for adaptive image contrast increasing // Turkish Journal of Computer and Mathematics Education Vol.12 No.10(2021), 5042-5045. DOI: <https://doi.org/10.17762/turcomat.v12i10.5278>
48. Орловский С.А. Проблемы принятия решений при нечеткой исходной информации. -М: Наука. 1981. 203с.

49 Подиновский В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. -М.: Наука. 1982.

50. Bodyanskiy Ye. An adaptive learning algorithm for a neuro-fuzzy network // Computational Intelligence. Theory and Applications / [Ed. By B. Reusch]. – Berlin; Heidelberg; New York: Springer, 2001. – P. 68–75.

51. Otto P. A new learning algorithm for a forecasting neurofuzzy network / P. Otto, Ye. Bodyanskiy, V. Kolodyazhniy / / Integrated Computer-Aided Engineering. – 2003. – 10. – № 4. – P. 399–409.