**Городской округ Краснотурьинск**

**Муниципальное автономное образовательное учреждение**

**«Средняя общеобразовательная школа № 24»**

**Направление: «Наука в цифрах»**

**Секция:**

**Проект**

**Искусственный интеллект. Модель, проектирование, разработка.**

**Автор проекта:**

***Самонин Евгений Игоревич****, 10 класс*

**Руководитель проекта:** *Ф.И.О., должность (с кем работаете)*

*телефон, эл. почта*

**2025**

**ОГЛАВЛЕНИЕ**

**Введение…**……………………………………………………………………………3

1. Теоретическая часть………………………………………………………………5
   1. Ключевые понятия………………………………………………………………5
   2. История создания нейронных сетей……………………………………………6
   3. Устройство нейронных сетей…………………………………………………...8

**Введение**

Искусственный интеллект каждый день берет новые вершины во всех сферах человеческой жизни. Он находит применение как в бытовой сфере, так и в производстве, образовании, медицине и многих других областях. Развитие и внедрение ИИ в нашу жизнь позволяет повысить эффективность работы сотрудников и компаний, улучшить качество предоставляемых услуг и оптимизировать многие сферы производства, а также увеличить собственный комфорт и понизить нагрузку на себя в тех вопросах, где вмешательство человека не обязательно.

Проектирование и разработка искусственного интеллекта в современности — одна из наиболее быстро развивающихся и перспективных областей как бизнеса, так и науки. В реалиях скачка компьютерных технологий и глобализации ИИ становится универсальным помощником для достижения преимуществ перед конкурентами и инноваций в различных направлениях экономики.

ИИ включает в себя огромное количество технологий, включая машинное обучение, обработку языка, компьютерное зрение, робототехнику и многое другое. Эти технологии и методы позволяют создавать системы, способные анализировать большие объемы данных и оперировать ими, принимать решения и обучаться на основе полученного опыта.

Для быстрого развития и внедрения ИИ требуются квалифицированные специалисты в области математики, информатики, программирования, статистики и других наук. Но, к сожалению, сейчас мало людей понимают устройство нейронных сетей и искусственного интеллекта.

**Проблема**

Большинство людей знают об искусственном интеллекте, но не знают, как он устроен и как он создается. Так проблема этого проекта состоит в том, что при огромном влиянии ИИ на нашу жизнь, большинство людей все равно им не пользуются и не понимают всей выгоды, что он несет для них.

**Обоснованность темы**

Тема "искусственный интеллект и его интеграция в нашу жизнь" обоснована, обоснована тем, что искусственный интеллект становится все более важным элементом нашего общества. Он применяется в различных сферах и полезен в жизни обычных людей, не связанных с ИИ.

**Актуальность работы**

Эта тема актуальна в современных условиях стремительного технологического прогресса. Разработка ИИ сейчас является большой сферой экономики и науки, в последние годы развитие ИИ стало появляться в заголовках новостей почти каждый месяц. Искусственный интеллект уже сегодня меняет нашу повседневную жизнь. С его помощью решаются сложные задачи, требующие анализа больших объёмов данных и автоматизации процессов. Интеграция ИИ способствует повышению эффективности и снижению человеческих ошибок. В образовательной сфере ИИ открывает новые возможности для личного обучения и повышения качества образовательных программ.

**Гипотеза**

Использование искусственных нейронных сетей позволяет автоматизировать задачи классификации данных, что повышает точность и скорость обработки информации по сравнению с ручными методами.

**Цель**

Я поставил перед собой цель создать модель нейронной сети. Для этого я поставил перед собой **задачи**:

* Изучить историю создания искусственного интеллекта.
* Проанализировать современные нейронные сети.
* Рассмотреть алгоритм работы полносвязного перцептрона.
* Создать свою модель нейронной сети.

1. **Теоретическая часть**
   1. **История создания нейронных сетей**

Изначально люди пытались воссоздать работу мозга человека с помощью компьютера и для этого выдвинули модели нейронов и связей между ними:

В 1943-м Уорреном Мак-Каллоком и Уолтером Питтсом была предложена математическая модель нейрона, а уже в конце 1950-х Фрэнк Розенблатт представил персептрон – простую модель машинного обучения, созданную для помощи компьютерам в обучении на разных объёмах данных. Именно его можно назвать первой практической реализацией нейросети. [4]

Именно для этого сначала использовать бинарная система активации нейрона. Либо нейрон активирован, либо нет. После этого многие поняли ограниченность нейронных сетей и настала «Зима искусственного интеллекта». Но позже с ростом сложности алгоритмов нейронных сетей, а также вычислительных возможностей компьютеров интерес появился вновь. В это время начали появляться способы «глубокого» обучения сетей, например обратное распространение ошибки, этот способ будет рассмотрен в этом проекте позже.

В период 1980-2000 стали разрабатываться и появляться первые алгоритмы обучения, сравнения и анализа данных, в XXI направление стало стремительно развиваться. В 2000-х появились мощные графические процессоры, и стали доступны большие объёмы данных, что побудило сообщество начать разработку алгоритмов Deep Learning – совокупность методов машинного обучения, основанных на обучении представлениям, а не на специализированных алгоритмах под конкретные задачи. [4]

В современном времени бум популярности нейронных сетей пришел с выходом Chat-GPT 3, который был способен осознанно отвечать на вопросы пользователя. После чего каждая крупная компания стала ставить перед собой цель конкурирования на рынке искусственного интеллекта.

* 1. **Современные нейронные сети**

Благодаря современным вычислительным мощностям, новым архитектурам и количеству данных для обучения нейронные сети стали не только машинами для предсказания и классификации, но и помощниками для обычных людей. В этом блоке будут рассмотрены современные нейронные сети, которые могут помогать обывателю без знания их устройства и примеры использования нейронных сетей крупными компаниями.

ИИ для обработки текста:

* Первооткрыватель и главный двигатель прогресса в мире ИИ — Chat-GPT. Мощнейший искусственный интеллект способный обрабатывать естественный язык, аудио и видео контент. Он способен сделать за вас текст любой стилистики и на любую тему.
* Google Gemini.
  1. **Понятие нейронной сети в современном мире**

Рассмотрим определение термина нейронной сети и ИИ.

Иску́сственный интелле́кт (ИИ) (англ. artificial intelligence; AI) в самом широком смысле — это интеллект, демонстрируемый машинами, в частности компьютерными системами. Это область исследований в области компьютерных наук, которая разрабатывает и изучает методы и программное обеспечение, позволяющие машинам воспринимать окружающую среду и использовать обучение и интеллект для выполнения действий, которые максимально увеличивают их шансы на достижение поставленных целей. Такие машины можно назвать искусственным интеллектом. ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие простые по отдельности процессоры вместе способны выполнять довольно сложные задачи [1].

Нейронная сеть (также искусственная нейронная сеть, ИНС, или просто нейросеть) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др. [2].

* 1. **Ключевые для понимания определения**

СПР — сеть прямого распространения.

Искусственный нейрон или просто нейрон — это составляющая часть слоя нейронной сети, обладающая связями (синапсами) с нейронами другого слоя. В нейрон входит сумматор и числовое значение нейрона (смещение).

Синапс — связь между нейронами разных слоев ИНС, имеющая вес.

Сумматор — это часть нейрона, суммирующая сигналы, поступающие от нейронов прошлого слоя через синапсы.

Полносвязный перцептрон — искусственная нейронная сеть, состоящая из полносвязных слоев.

Гиперпараметры нейросети — это параметры, которые не меняются в процессе обучения нейросети и подбираются вручную разработчиком исходя из вычислительных мощностей и потребностей. В нейросети это количество скрытых слоев, количество нейронов в скрытых слоях.

Весы ИНС — все весы синапсов и смещения (все обучаемые характеристики)

Батч (batch) — неполная подвыборка из всего массива данных.

1.1 Понятие нейронной сети в современном мире

Рассмотрим определение термина нейронной сети и ИИ.

Иску́сственный интелле́кт (ИИ) (англ. artificial intelligence; AI) в самом широком смысле — это интеллект, демонстрируемый машинами, в частности компьютерными системами. Это область исследований в области компьютерных наук, которая разрабатывает и изучает методы и программное обеспечение, позволяющие машинам воспринимать окружающую среду и использовать обучение и интеллект для выполнения действий, которые максимально увеличивают их шансы на достижение поставленных целей. Такие машины можно назвать искусственным интеллектом. ИНС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие простые по отдельности процессоры вместе способны выполнять довольно сложные задачи [1].

Нейронная сеть (также искусственная нейронная сеть, ИНС, или просто нейросеть) — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др. [2].

1.2 Ключевые для понимания определения

СПР — сеть прямого распространения.

Искусственный нейрон или просто нейрон — это составляющая часть слоя нейронной сети, обладающая связями (синапсами) с нейронами другого слоя. В нейрон входит сумматор и числовое значение нейрона (смещение).

Синапс — связь между нейронами разных слоев ИНС, имеющая вес.

Сумматор — это часть нейрона, суммирующая сигналы, поступающие от нейронов прошлого слоя через синапсы.

Полносвязный перцептрон — искусственная нейронная сеть, состоящая из полносвязных слоев.

Гиперпараметры нейросети — это параметры, которые не меняются в процессе обучения нейросети и подбираются вручную разработчиком исходя из вычислительных мощностей и потребностей. В нейросети это количество скрытых слоев, количество нейронов в скрытых слоях.

Весы ИНС — все весы синапсов и смещения (все обучаемые характеристики)

Батч (batch) — неполная подвыборка из всего массива данных.

Стохастический градиентный спуск (англ. Stochastic gradient descent, SGD) — итерационный метод для оптимизации целевой функции с подходящими свойствами гладкости (например, дифференцируемость или субдифференцируемость). Его можно расценивать как стохастическую аппроксимацию оптимизации методом градиентного спуска, поскольку он заменяет реальный градиент, вычисленный из полного набора данных, оценкой, вычисленной из случайно выбранного подмножества данных. Это сокращает задействованные вычислительные ресурсы и помогает достичь более высокой скорости итераций в обмен на более низкую скорость сходимости. Особенно большой эффект достигается в приложениях, связанных с обработкой больших данных [3].

* 1. **Устройство нейронных сетей**

1. В этом проекте будет рассматриваться нейронная сеть — полносвязный перцептрон.

**1.3.1 Основные выполняемые нейросетями функции**

Классификация — распределение данных по классам на основе определённых параметров. Например, на вход дается набор характеристик человека и от нейросети требуется понять: давать человеку кредит или нет.

Предсказание — возможность анализировать закономерности и на их основе давать предсказания. Например, рост или падение акций, основываясь на ситуации на фондовом рынке.

Распознавание — Идентификация и детектирование объектов на изображении.

В этом проекте я буду рассматривать устройство классифицирующей нейросети для дальнейшего ее создания.

**1.3.2 типы слоев ИНС**

Входной — нейроны, на которые изначально подается входная информация, будь это слова текста или пиксели фотографии.

Скрытые — нейроны, находящиеся между входными и выходными, слоев этих нейронов может быть сколько угодно. Пользователь никак не контактирует с этими нейронами, поэтому они и называются скрытыми. Количество скрытых нейронов во внутренних слоях может меняться.

Выходные нейроны — последний слой нейронов, дающий окончательный результат.

Рассмотрим каждый тип подробнее:

Входной слой всегда один. Входных нейронов в слое должно быть столько же, сколько у нас есть входных данных. Например, если на вход подается изображение разрешением 28 на 28 пикселей и каждый пиксель закодирован одним числом, то входных нейронов должно быть 28 \* 28 = 784. Значение входного нейрона никак не обрабатывается и передается следующим слоям неизменным.

Скрытых слоев может быть сколько угодно (гиперпараметр нейросети), количество нейронов в каждом слое тоже гиперпараметр. В классифицирующих нейросетях в идеальном случае нейроны каждого последующего слоя должны отвечать за более общую картину. Например, в нейросети, классифицирующей изображение цифры, в первом слое (входном) каждый нейрон отвечает за один пиксель, нейроны второго слоя — за отдельные черты, третьего — за отдельные элементы цифры, и после в выходном слое всё это сводится до одного ответа, какая это цифра. Нейроны скрытых слоев берут свое значение исходя из значений всех нейронов предыдущего слоя и передают его на следующий слой.

Выходной слой тоже только один. Нейронов в этом слое столько же, сколько возможных вариантов ответа. Ответом является нейрон с наибольшим значением.

**1.3.3 Вычисление результата работы нейронной сети зная весы**

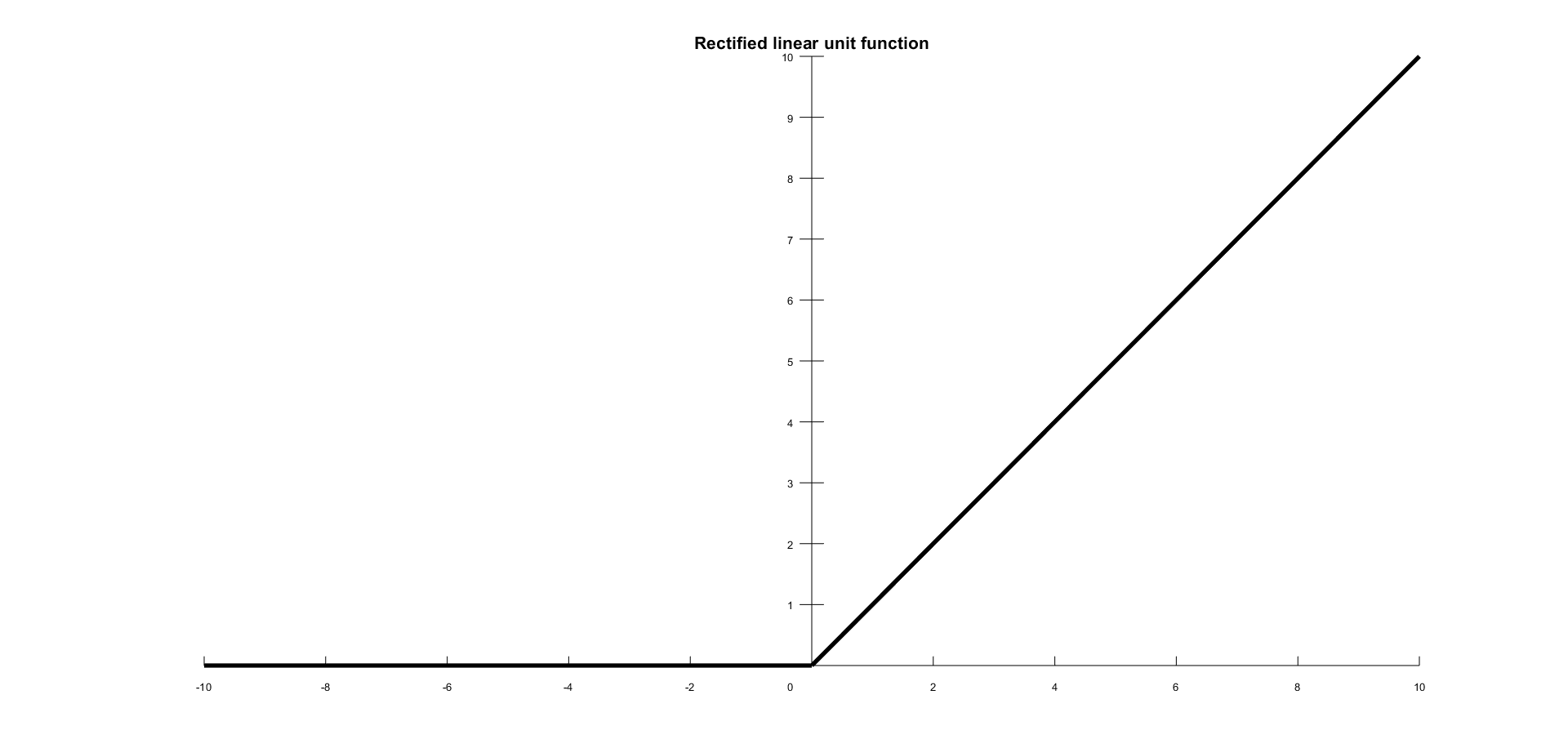
Уже рассмотрено, какие есть слои и что примерно делают нейроны в этих слоях. Пришло время разобраться, как они это делают.

Начнем с того, как нейрон преобразует переданную ему информацию. Как стало понятно из вышесказанного, нейроны входного слоя ничего не делают с этой информацией, тогда нужно понять, как это делают нейроны скрытых и выходного слоев.

Синапс — это связь между двумя нейронами, которая имеет вес. У каждого нейрона одного слоя есть синапсы с каждым нейроном предыдущего и последующего слоя. И так каждый нейрон передает свое значение каждому следующему нейрону, умножая его на вес синапса конкретного нейрона. Эта информация поступает на нейрон и суммируется с также полученными значениями других нейронов, после чего к этой сумме прибавляется смещение нейрона. Дальше эту сумму нужно нормализовать, для этого и используется функция нормализации, в моем случае я буду использовать функцию ReLU.

ReLU (rectified linear unit) — это функция, превращающая отрицательные значения в 0, а положительные оставляет как есть.

Формула ReLU:

И ее график:

Для этого и нужно смещение, чтобы задать больше, чего должна быть сумма значений предыдущих нейронов, например, чтобы нейрон активировался, только если сумма больше 10 или больше -100.

После нормализации значения сигнал передается дальше по синапсам в следующие слои.

Рассмотрим всю эту схему на примере с двумя входными нейронами и одним скрытым нейроном. Обозначим входные нейроны I1, I2, скрытый как N, веса синапсов как W и смещение B. Тогда входными и выходными данными на нейрон будут:

*Ninput = I1\*W1 + I2\*W2*

*Noutput = ReLU(I1\*W1 + I2\*W2 + B)*

Запишем всё это в алгебраическом виде и сделаем формулы общими. Представим n-ный (отсчет идет с нуля) нейрон L-ого слоя (кроме входного и выходного) нейронной сети, обозначим ALn, нейроны предыдущего слоя AL-1k, весы синапсов как Wk, n, а смещение как Bn, а, где k— это номер нейрона предыдущего слоя, а n— это номер рассматриваемого нейрона. Значение этого нейрона: A = ReLU (W0,0AL-10 + W1,0AL-11+ ... + Wn, 0\*AL-1k + Bn). Если рассмотреть эту формулу без смещения и ReLU поподробнее, можно понять, что это первый элемент вектора, получаемого при умножении вектора значений нейронов предыдущего слоя и матрицы весов (каждая строка — это вектор весов синапсов нейрона предыдущего слоя со всеми нейронами текущего) = W0,0\*A0 + W0,1\*A1 + ... + W0,n\*An. Тогда, чтобы получить наше значение, нужно к этому произведению прибавить вектор смещений и взять сигмоиду. И все это можно красиво записать для первого слоя: A1 = ReLU(WA0 + B) или в общем виде:

L – номер слоя, n – количество нейронов в новом слое, k – количество нейронов в предыдущем слое.

Так применяя эту формулу для каждого слоя, нейросеть доходит до последнего, а в нем нужно использовать другую функцию. Функция Softmax преобразует данный ей вектор в вектор вероятностей, так чтобы сумма этих вероятностей была равна единице. Её формула для каждого члена вектора:

Так ответом нейросети является класс, у которого наибольшая вероятность.

**1.3.4 Обучение нейронной сети**

Нейросеть учится на данных. Эти данные должны быть заранее заготовлены и сгруппированы. Дальше эти данные делятся на два типа: тренировочные и тестовые. Тренировочные используются напрямую при тренировке. Тестовые используются для отслеживания динамики обучения и в конце для оценки эффективности работы ИНС.

Обучение нейронной сети происходит по этапам, каждый этап называется эпохой. Данные для обучения для каждой эпохи одинаковые, и эпохой и называется одна итерация обучения на базе данных. Для каждой эпохи желательно каждый раз перемешивать данные. В каждой эпохе происходят следующие действия:

1. Деление данных на батчи (размер батча — это гиперпараметр).
2. Прямое распространение, т. е. получение результатов предсказания нейросети для этих данных.
3. Обратное распространение ошибки — это способ получения вектора градиента.
4. Корректировка значений весов на основе суммарной ошибки за весь батч.

И далее те же действия по итерациям батчей и эпох.

Рассмотрим обучение способом стохастического градиентного спуска. Для осуществления градиентного спуска нужно получить градиент функции ошибки. Функция ошибки — это функция, показывающая насколько ошиблась нейросеть. А градиент — это вектор, каждая координата, которого показывает на сколько зависима функция ошибки от значения этой переменной и, следовательно, на сколько нужно изменить соответствующий вес (следовательно, координат у него столько же сколько весов и смещений у нейросети), и направление которого указывает, как функция будет возрастать с наибольшей скоростью. Так как это градиент ошибки, а ее необходимо минимизировать, то программа вычитает из весов его значение, соответствующий этому весу. Получить этот вектор можно с помощью обратного распространения ошибки. Формулы приведены ниже.

Пусть L это номер слоя, T это транспонированная версия вектора или матрицы, WL это матрица весов между слоями L и L – 1, BL это вектор смещений данного слоя, Z то, что передается в функцию активации, F это функция активации, ⊙это поэлементное умножение, AL это вектор значений слоя L. То на сколько нужно изменить вес обозначается производной ошибки по этому весу, но я буду рассматривать формулы относительно матриц и векторов целых слоев для большей производительности программы.

После чего становится для предыдущего слоя и все повторяется по итерациям. Но для первой итерации нужно знать последнего слоя. Для этого есть отдельная формула. Для нее введём еще несколько обозначений. Y это истинный результат, SoftMax(Zmax) = P (наш результат работы нейросети в вероятностях):

Но так как нейронная сеть работает не с единичными прогнозами, а с батчами, то формулы изменятся. Из-за свойств матричного умножения почти все формулы останутся теми же, кроме одной (сумма производной ошибки по Z за весь батч):

A так же вычислим саму ошибку для отслеживания прогресса обучения нейросети. Для этого воспользуемся функцией CrossEntropy(P, Y) = . Но так как в истинном распределении вероятности только одна единица и остальные нули и Y это номер элемента правильного ответа, от суммы останется только

После рассмотрения принципа обучения нейросети, нужно научиться делать это в меру, так как ИНС может переобучиться. Это значит, что она слишком хорошо запомнит тренировочные данные и станет хуже работать на тестовых данных. Оптимальные гиперпараметры для конкретной нейросети подбираются в ходе ее обучения по валидационным данным.

1. **Практическая часть**

**Задача**

Реализовать полносвязный перцептрон на языке Python без использования заготовленных библиотек вроде TensorFlow, дающих возможность создавать нейронные сети без написания их логики работы. И сделать это парадигме ООП, то есть создать класс нейросети для дальнейшей возможности импортирования этого класса в другие программы и большей универсальности.

**Инструментарий**

Язык программирования Python. библиотеки: numpy (np) — для реализации матричных математических операций, Данные: база данных MNIST (60 тыс. тренировочных и 10 тыс. тестовых изображений цифр в формате 28 \* 28 пикселей).

**2.2 Код программы**

При обучении, необходимо инициализировать веса и смещения случайными значениями, например, центрированными около нуля и со средним отклонением, равным квадратному корню из количества нейронов в слое. И прежде, чем реализовывать обучение нейросети, нужно создать функции нужные для этого.

def init\_weights(self, layers):

    self.weights = []  # Инициализация пустых весов

    self.bias = []

    for num in range(len(layers) - 1):  # Присваивание случайных значений для всех слоев

        self.weights.append(

            np.random.normal(

                0.0, pow(layers[num], -0.5), (layers[num], layers[num + 1])

                )

            )

        self.bias.append(

            np.random.normal(

                0.0, pow(layers[num + 1], -0.5), (layers[num + 1])

                )

            )

def ReLU(x):  # Функция активации

    return np.maximum(0, x)

def ReLU\_deriv(Z):  # Производная ReLU

    return np.where(Z > 0, 1, 0)

def SoftMax\_batch(vect):  # Преобразование батча к вероятностям

    exp\_vect = np.exp(vect - np.max(vect, axis=1, keepdims=True))

    return exp\_vect / np.sum(exp\_vect, axis=1, keepdims=True)

def cross\_entropy\_batch(p, y, epsilon=1e-12):  # Вычисление расстояния между вероятностями

    p = np.clip(p, epsilon, 1.0)  # Защита от деления на 0

    return -np.sum(np.log(p[np.arange(len(y)), y])) / len(y)

# Преобразование вектора правильных ответов к матрице истинных вероятностей

def y\_vector\_batch(y, num\_classes):

    return np.eye(num\_classes)[y]

def num\_correct(p, y): # Подсчет верных ответов

    return np.sum(np.argmax(p, axis=1) == y)

def predict\_batch(self, data):  # Предсказание нейросети для батча

    for i in range(len(self.bias) - 1):

        data = ReLU(data @ self.weights[i] + self.bias[i])

    return SoftMax\_batch(data @ self.weights[-1] + self.bias[-1])

def test(self, data, labels):

    return num\_correct(self.predict\_batch(data), labels) / data.shape[0] \* 100

def num\_correct(p, y): # Подсчет верных ответов

    return np.sum(np.argmax(p, axis=1) == y)

После каждой эпохи обучения выводим cross entropy по всей обучающей выборке, точность на обучающей и тестовой выборках для отслеживания прогресса.

Пришло время тренировать нейросеть. Количество слоев выберем 3, количество нейронов в слоях: 784, 256, 10, размер батча 5000, скорость обучения 0.0001 и 50 эпох.

def learn(self, data, labels, epochs, batch\_size, train\_rate, valid\_data, valid\_labels):

    print('Начало обучения')

    for ep in range(epochs):

        epoch\_loss = 0

        correct\_count = 0

        indices = np.arange(data.shape[0])  # Перемешивание данных

        np.random.shuffle(indices)

        data = data[indices]

        labels = labels[indices]

        for i in range(data.shape[0] // batch\_size):

            # Прямое распространение нейросети

            layer = [data[i \* batch\_size : (i + 1) \* batch\_size]]

            y = labels[i \* batch\_size : (i + 1) \* batch\_size]

            Z = []

            for L in range(len(self.weights) - 1):

                Z.append(layer[L] @ self.weights[L] + self.bias[L])

                layer.append(ReLU(Z[L]))

            layer.append(SoftMax\_batch(layer[-1] @ self.weights[-1] + self.bias[-1]))

            # Обратное распространение

            dE\_dZ = layer[-1] - y\_vector\_batch(y, self.weights[-1].shape[1])

            for L in range(len(self.weights) - 1, -1, -1):

                dE\_dW = np.transpose(layer[L]) @ dE\_dZ

                dE\_dB = np.sum(dE\_dZ, axis=0, keepdims=True)

                dE\_dB.resize(self.bias[L].shape)

                if L > 0:

                    dE\_dA = dE\_dZ @ np.transpose(self.weights[L])

                    dE\_dZ = dE\_dA \* ReLU\_deriv(Z[L - 1])

                # Корректировка весов

                self.weights[L] -= train\_rate \* dE\_dW

                self.bias[L] -= train\_rate \* dE\_dB

            epoch\_loss += np.sum(cross\_entropy\_batch(layer[-1], y))

            correct\_count += num\_correct(layer[-1], y)

        # Вывод номера эпохи и точности нейросети

        accuracy = correct\_count / data.shape[0] \* 100

        print(f'Epoch {ep+1}/{epochs}: Loss = {epoch\_loss:.2f}, Train accuracy = {accuracy:.2f}%, Valid accuracy = {self.test(valid\_data, valid\_labels):.2f}%')

    print(f'Нейросеть обучена')

digit = Network()

digit.init\_weights([784, 256, 10])

digit.learn(train\_data, train\_labels, 25, 5000, 0.0001, test\_data, test\_labels)

digit.save(r'digit')

По выводу в терминале видим, что погрешность уменьшается:

Начало обучения

Epoch 1/50: Loss = 16.73, Train accuracy = 62.22%, Valid accuracy = 73.51%

Epoch 2/50: Loss = 8.26, Train accuracy = 78.98%, Valid accuracy = 82.88%

Epoch 3/50: Loss = 6.61, Train accuracy = 82.63%, Valid accuracy = 86.87%

…

Epoch 34/50: Loss = 1.75, Train accuracy = 95.89%, Valid accuracy = 95.72%

…

Epoch 49/50: Loss = 1.38, Train accuracy = 96.84%, Valid accuracy = 96.35%

Epoch 50/50: Loss = 1.35, Train accuracy = 96.88%, Valid accuracy = 96.45%

Нейросеть обучена

Финальная точность вышла 96.45%. Если загрузить нейросеть и пытаться дальше ее обучать, то видно, что погрешность не уменьшается или даже растет, это значит, что нейросеть переобучилась.

Полный код нейросети размещен на GitHab [[5]](https://github.com/EvgeniySamon/neural_network).

**Заключение**

2. Практическая значимость

Созданная модель искусственной нейронной сети может быть использована для образовательных целей, обучения основам машинного обучения, а также для разработки практических приложений, таких как системы распознавания изображений, что упрощает внедрение технологий ИИ в образовательную и исследовательскую деятельность.

3. Продукт

Продуктом является класс нейронной сети, реализующий модель полного перцептрона на языке Python. Эта реализация может быть использована для обучения нейросети на различных наборах данных, включая MNIST, что демонстрирует возможности ИИ в решении задач классификации.

**Библиография**

1. Определение искусственного интеллекта. <https://ru.wikipedia.org/wiki/ИИ>
2. Определение нейронной сети. <https://ru.wikipedia.org/wiki/Нейронная_сеть>
3. Определение стохастического спуска. <https://ru.wikipedia.org/wiki/Стохастический_градиентный_спуск>
4. История создания нейронных сетей. <https://habr.com/ru/articles/861888/>
5. Репозиторий проекта на Githab. <https://github.com/EvgeniySamon/neural_network>