

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Кафедра

вычислительной техники

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

**ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ
ПРИ ОБУЧЕНИИ ВЫСШЕЙ МАТЕМАТИКЕ
В ВУЗЕ**

Студент

Leo 04.06.2024
(подпись, дата)

Леонова Татьяна Юрьевна
(ФИО полностью)

Руководитель

Art 04.06.2024
(подпись, дата)

Артюхин В.В.
(фамилия, инициалы)

Нормоконтролёр

главн. естествозн. 04.06.2024
(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Рецензент

А.О. КИП «Рубин» 4 04.06.2024
(должность, место работы) (подпись, дата)

Васильков А.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой

(подпись)

Митрохин М.А.
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

21.06.2024 № 36)

Секретарь ГЭК

(подпись)

Гурин Е.И.
(фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«13» сентября 2022 г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

1. Студент Леонова Татьяна Юрьевна гр. 22ВВИм2 факультета
вычислительной техники, направления 09.04.01

2. Тема работы «Использование нейронных сетей при обучении высшей математике
в вузе»

Тема утверждена приказом ПГУ № 765/о от «13» сентября 2022 г.

3. Руководитель магистерской диссертации к.т.н., доцент Артюхин В.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Назначение разработки: определение уровня знаний студента для дальнейшей модернизации учебного процесса, с целью поднятия интеллектуального уровня студентов.

Исходные данные: результаты тестирования, которые были получены при решении студентами разноуровневых заданий.

Создать, обучить и внедрить нейронную сеть, с целью определения уровня обученности студентов при изучении высшей математики.

5. Перечень подлежащих разработке вопросов

1. Анализ предметной деятельности нейронных сетей и их возможностей при обучении высшей математики.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области дифференцированного подхода в обучении с помощью компьютерного тестирования	10%	13.09.2022 - 21.11.2022	<i>Арт</i>
2	Анализ предметной области использования нейронных сетей в образовательном процессе	10%	22.11.2022 - 22.01.2023	<i>Арт</i>
3	Анализ предметной области нейронных сетей и их обучения	20%	23.01.2023 - 23.04.2023	<i>Арт</i>
4	Разработка нейронной сети для определения уровня обученности студентов	40%	24.04.2023 - 29.01.2024	<i>Арт</i>
5	Тестирование разработанной нейронной сети, отладка	10%	30.01.2024 - 30.03.2024	<i>Арт</i>
6	Оформление выпускной работы	10%	31.03.2024 - 31.05.2024	<i>Арт</i>

Дата выдачи задания «13» сентября 2022 г.

Руководитель магистерской диссертации *Арт* 13.09.22 Артюхин.В.В.
(подпись, дата)

Задание к исполнению принял студент *Лео* 13.09.22 Леонова Т.Ю.
(подпись, дата)

Работу к защите допустить
Декан факультета *Фео* 20.06.2024 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

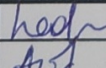
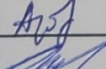
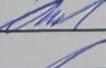
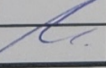
Магистерская диссертация содержит 83 листа, 19 рисунков, 3 таблицы, 36 источников.

УРОВНЕВАЯ ДИФФЕРЕНЦИАЦИЯ, КОМПЬЮТЕРНОЕ ТЕСТИРОВАНИЕ, НЕЙРОННЫЕ СЕТИ

Цель работы состоит в повышении эффективности процесса обучения математике в вузе с помощью использования нейронной сети, которая сможет определить уровень знаний студентов по результатам тестирования при изучении высшей математики. Объектом разработки является процесс обучения математике студентов технических вузов.

В результате проведенной работы разработана, обучена и протестирована нейронная сеть, которая определяет уровень обученности студентов на основе результатов тестирования, учитывалось время выполнения задания и количество ошибок, которые допустил обучающийся за время выполнения того или иного задания.

Результаты данной работы могут быть применены в образовательном процессе, а также для дальнейших научных исследований в усовершенствовании педагогического процесса при использовании технических достижений.

					ПГУ1.090401.51.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Использование нейронных сетей при обучении высшей математике в вузе Пояснительная записка	Лит.	Лист	Листов
Разраб.		Леонова Т.Ю.		04.06.24				
Провер.		Артюхин В.В.		04.06.24			2	83
Н. Контр		Бычков А.С.		04.06.24		ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		04.06.24		гр. 22ВВИм2		

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1 Теоретический анализ предметной области.....	6
1.1 Методы определения уровня обученности в традиционной системе образования.....	6
1.2 Компьютерное тестирование как средство определения уровня обученности студентов.....	15
1.3 Анализ возможностей использования нейронных сетей при обучении математике	21
2 Разработка нейросетевой модели для определения уровня обученности студента при изучении высшей математики	29
2.1 Набор данных для обучения нейронной сети	29
2.2 Выбор архитектуры нейронной сети	33
2.3 Описание разработанной НС	42
2.4 Описание разработанной программы	49
ЗАКЛЮЧЕНИЕ	64
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	65
ПРИЛОЖЕНИЕ А Листинг программы	69
ПРИЛОЖЕНИЕ Б Презентация.....	78

ВВЕДЕНИЕ

В силу стремительных изменений, происходящих в мире, все больше растет потребность в специалистах различных сфер деятельности человека. Справедливо заметить, что темпы развития образования отстают от скорости развития общества. Получение актуального и качественного образования будущими специалистами является одной из главных проблем современного вуза, при этом большинство преподавателей университетов понимают, что без внедрения достижений технического прогресса проблему качества образования не решить.

В следствии этого, появляется необходимость в создании эффективных средств анализа полученной информации, связанной с различными аспектами и этапами обучения, в частности при изучении высшей математики в вузе, а именно результатов обучения, для дальнейшего использования этих данных с целью повышения уровня образования каждого студента.

Реализация возможностей технических средств, в частности, искусственного интеллекта, поможет усовершенствовать учебный процесс, таким образом, чтобы эта проблема разрешилась. При введении нейронных сетей в процесс обучения, вуз получает возможность оценивать: качество подготовки профессорско-педагогического состава, качество полученных знаний студентами с целью дальнейшей ликвидации пробелов в различных направлениях образовательного процесса, состояние материально-технической базы университета, уровень конкурентоспособности специалистов на рынке труда.

Анализируя вышесказанное можно выделить сложившуюся в настоящее время потребность совершенствования учебного процесса за счет использования современных цифровых технологий. Наличие данной проблемы обуславливает актуальность и выбор темы исследования.

Научная проблема работы состоит в разработке методов и средств совершенствования математической подготовки студентов технических специальностей на основе внедрения нейронных сетей в процессе обучения.

Цель исследования состоит в повышении эффективности процесса обучения математике в вузе с помощью использования нейронных сетей.

Объект исследования – процесс обучения математике студентов технических вузов.

Предмет исследования – возможности применения некоторых свойств искусственного интеллекта для модернизации процесса обучения в вузе.

Достижение поставленной цели предполагает решение следующих задач:

- на основе анализа основных понятий нейронных сетей и их классификации, выделить возможности их использования в учебном процессе,
- обоснование выбора внедрения и использования возможностей нейронных сетей с целью повышения качества образовательного процесса,
- разработка системы, позволяющей использовать нейронные сети и их возможности при обучении математике.

Практическая ценность работы.

Внедрение нейронных сетей в обучении математике позволит проанализировать такие основополагающие направления деятельности вуза как:

- ведение учебного процесса (методы и формы обучения, стимулирование и мотивация, цель и результат),
- управление профессорско-педагогическим составом кафедры,
- научно-исследовательская, инновационная работа,
- обеспечение учебно-методическими материалами,
- социальное и техническое обеспечение.

Таким образом, использование нейронных сетей при обучении математике в вузе имеет важное значение в управлении и развитии высшего учебного заведения для достижения наилучшего результата во всех вышеперечисленных аспектах.

1 Теоретический анализ предметной области

1.1 Методы определения уровня обученности в традиционной системе образования

В настоящий момент основная идея дифференцированного подхода обучения состоит в идее представления каждому обучающемуся полного образования, используя все их способности и достижения.

Одной из основных целей обучения в университете - предоставить полноценное образование каждому студенту учитывая его индивидуальные способности и интересы. Реализация данного подхода возможна при обеспечении обучающихся таких комфортных условий, при которых был бы реализован процесс обучения каждого студента в группе. Существует противоречие в традиционном образовании: студентов в группе много, но к каждому необходимо найти свой подход. Реализация дифференцированного подхода в учебном процессе поможет решить эту задачу.

Существует множество взглядов на такую проблему, как дифференциация обучения. Выделим одно из них: «дифференциация (различение, разделение целого на различные части, формы, ступени) - это...форма организации учебной деятельности школьников среднего и старшего возраста, при которой учитываются их склонности, интересы и проявившиеся способности [1]».

В данной магистерской диссертации предполагается дифференцировать студентов по их уровню знаний после изучения какой-либо темы, поэтому рассмотрим цели дифференциации (рисунок 1) более подробно, чтобы более эффективно и целесообразно построить педагогический процесс, учитывая все аспекты рассматриваемой проблемы.

В нашем исследовании речь идет об обучении в высших учебных заведениях, где ведущей выступает уровневая дифференциация. Это обосновано тем, что при выборе направления подготовки студент уже выбрал так называемый профиль обучения.

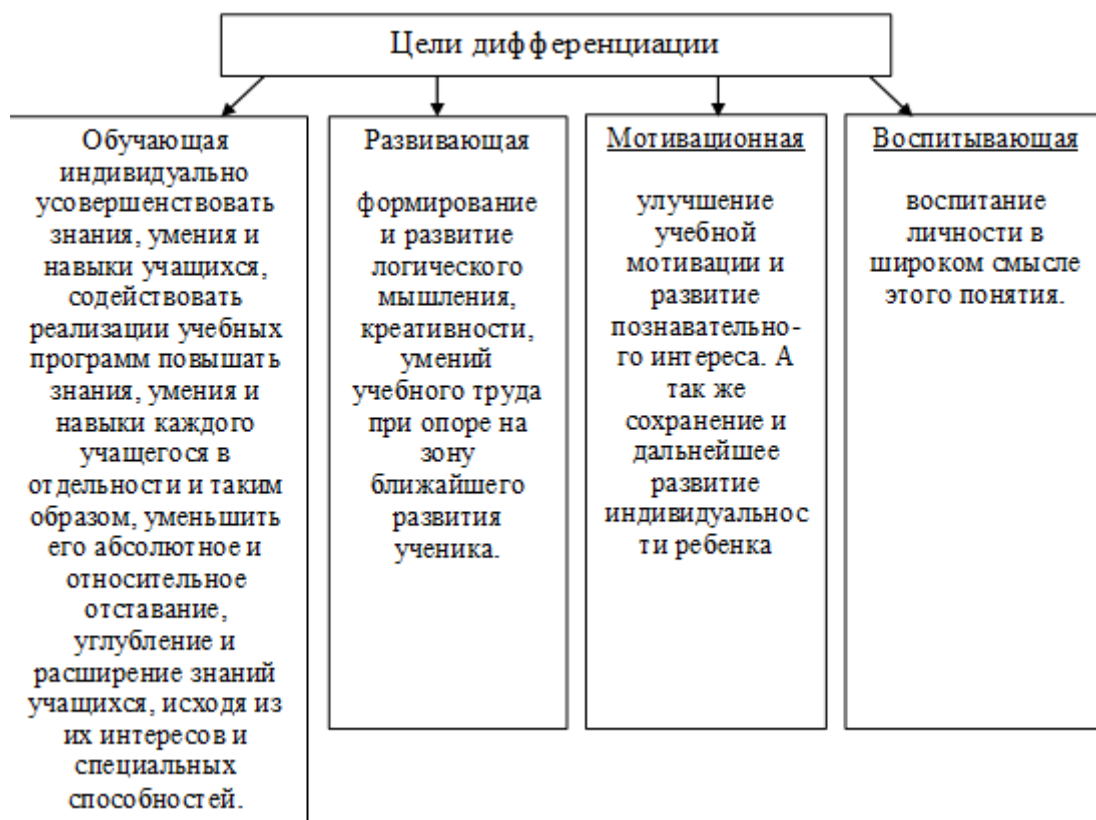


Рисунок 1 - Цели дифференциации

Реализуя цели данного исследования, предполагается вести обучения высшей математики, используя разноуровневый подход. Все уровни, рассматриваемые в данной работе, будут определяться на основе минимального (обязательного) уровня обученности. Студенты, поступившие в университет, определились с профилем и выбором направления, поэтому в силу проводимого исследования остановимся более подробно на таком виде дифференциации, как уровневая. При ее реализации все обучающиеся получают знания в одних условиях, но усвоение и дальнейшее применение полученных знаний идет строго индивидуально, так как будут учитываться многие показатели, например, такие как время выполнения задания, соответствующего уровня и количество ошибок, допущенных студентами при выполнении примеров.

Вообще говоря, уровневая дифференциация предполагает такую организацию обучения, при которой учащиеся, обучаясь по одной программе,

имеют право и возможность усваивать ее на различных планируемых уровнях, но не ниже уровня обязательных требований [2]. Следовательно, все уровни выше обязательного формируются на основе уровня обязательной подготовки.

Предполагается, что студенты будут получать знания по высшей математике с учетом их возможностей, в частности им будут предложены задачи различной степени сложности, учитывая их уровень подготовки и мыслительной деятельности, после отработки «пробелов» ставится задача перехода обучаемых на более высокий уровень, за счет пополнения недостающих знаний путем предоставления соответствующей литературы и консультаций непосредственно преподавателем, т.е. будет осуществлён комплексный подход к каждому студенту, что является одним из направлений данного исследования.

Следует отметить, что главной задачей уровневой дифференциации является организация образовательного процесса на таком уровне, чтобы он соответствовал уровню обученности конкретного студента, это позволит создать такие условия, в которых каждый обучающийся будет чувствовать себя максимально комфортно, что значительно повысит мотивацию к получению знаний и повысит заинтересованность в учебном предмете.

Проведем сравнительную характеристику положительных и отрицательных моментов дифференцированного подхода при изучении высшей математики в вузе (рисунок 2). Данный подход будем рассматривать в разрезе создания относительно стабильных или временных учебных групп, различающихся по тем или иным признакам (содержание, уровень учебных требований, интересы, форма обучения и т.п.) [3]. Первоначальное разделение на группы предполагается на осознанном и добровольном выборе самого обучаемого, дальнейшего разделение на группы будет происходить после тестирования по каждой теме. Целью данного исследования является повышение уровня до максимального, так как хорошие и стабильные знания

по математике существенно влияют на дальнейшее формирования специалистов технических вузов [2].

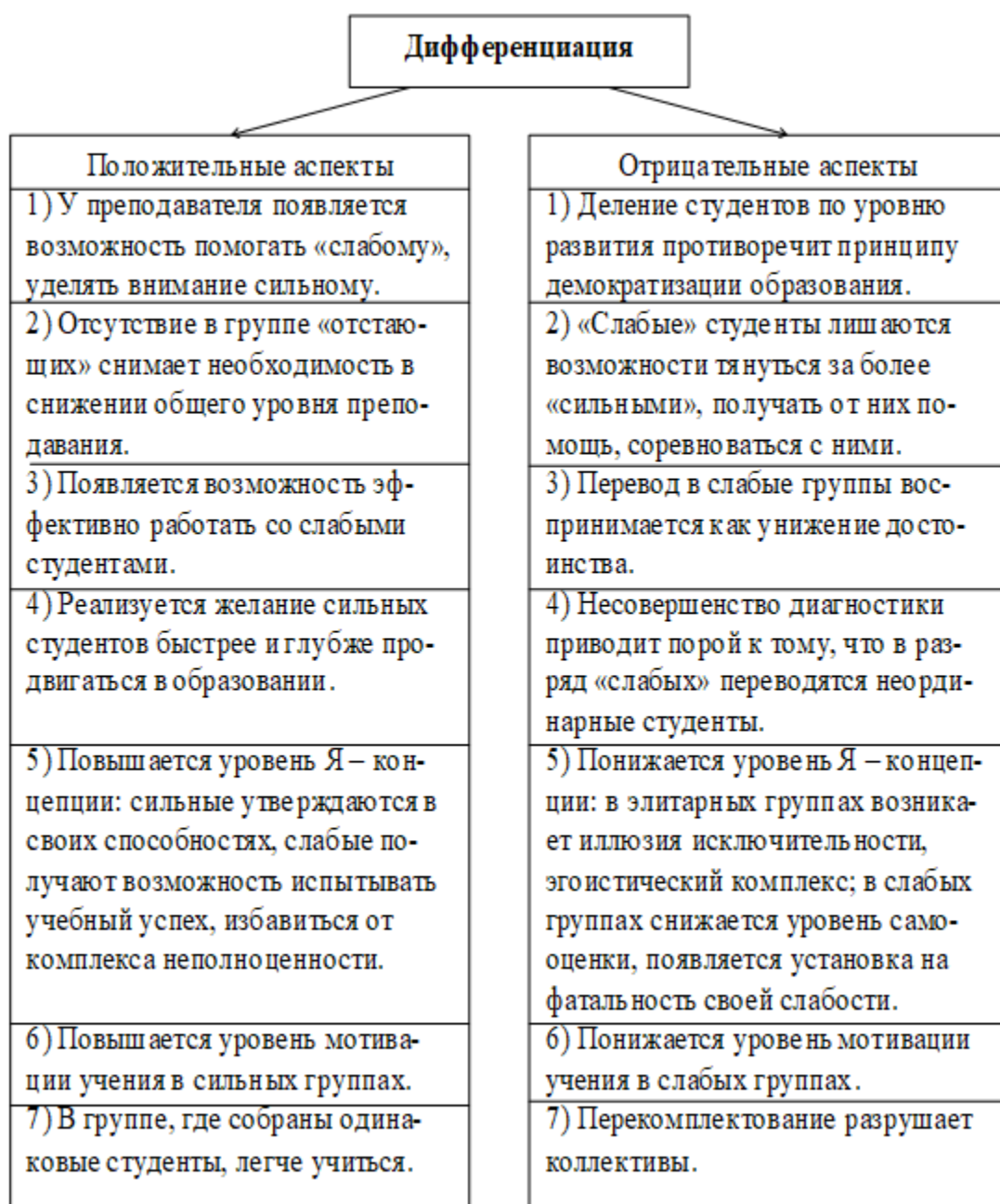


Рисунок 2 - Положительные и отрицательные аспекты дифференциации

Если далее вести разговор о выборе методики обучения студентов средних специальных или высших учебных заведений, то начальным этапом должно стать определение первоначального уровня компетенций в определенной области знаний: то есть преподаватель должен определить начальный уровень знаний и умений, а также ближайшую перспективу – что

студенту предстоит узнать. Также возможно определение того слоя знаний, которые возможно были забыты. Как известно из работ С.И. Архангельского закону преемственности учебных знаний [4], новые знания по любому учебному предмету должны возникать на основе уже известного обучаемым. Не является исключением и математика. В результате недостаточного усвоения учащимися так называемых промежуточных тем возникает сложность при дальнейшем обучении, а иногда и невозможность в усвоении новых понятий и нового материала. В данном случае нарушается закон преемственности знаний и последовательного научного развития, что существенно влияет на процесс обучения в целом. Студентам нужна мотивация и поддержка на каждом этапе получения новых знаний, особенно на первом и втором курсах, когда происходит изучение высшей математики.

Основным моментом при реализации принципов уровневой дифференциации является корректное определение первоначального уровня обученности студентов. В дальнейшем планируется для реализации этого принципиального момента использовать научно-технические достижения человека.

Реализация принципов уровневой дифференциации будет построена таким образом, что на основе выделенного уровня обязательной подготовки будут формироваться повышенные уровни изученной дисциплины. Процесс обучения будет построен таким образом, что каждый студент имел возможность получить знания в той мере и темпе, в котором ему будет комфортно, но уровень приобретаемых знаний должен быть не ниже обязательного (минимального) уровня знаний, который прописан в стандарте и учебном плане.

Применение дифференцированного подхода при обучении высшей математики с использованием достижений технического прогресса потребует

преобразования традиционных форм, средств и методов преподавания.

Перечислим некоторые недостатки при традиционной форме обучения:

- крайне редко при объяснении устного материала используется интерактивная доска или проектор,

- студенты при выполнении домашнего задания либо при выполнении самостоятельной работы не посещают библиотеки, то есть категорически мало используют в процессе обучения основную и дополнительную литературу,

- в силу уменьшения выделенных часов на изучение высшей математики у преподавателя иногда не хватает времени на проверку выполненного домашнего задания и на отслеживание как студент освоил промежуточные темы, проведение контрольных работ и устных опросов становится нецелесообразным,

- все задачи и предлагаемые для решения примеры в основных учебниках и учебных пособиях по высшей математике представлены без учета индивидуальных способностей каждого студента,

- удельный вес самостоятельной работы на практических занятиях крайне незначителен, а на лекциях отсутствует совсем.

Конечным результатом освоения той или иной дисциплины в университете считаем подготовкой студента к практической реализации своих знаний, чтобы обучаемый мог легко приспосабливаться к стремительным изменениям, происходящим в мире. Весь акцент процесса обучения должен быть смещен на развитие логической, мыслительной, познавательной, творческой, исследовательской деятельности студента.

Для решения данных проблем целесообразно в процесс обучения вводить новые технологии, а именно внедрять технические достижения при дифференцированном подходе. Было решено использовать возможности искусственного интеллекта для определения уровня знаний студентов для дальнейшей активации познавательной и умственной деятельности

обучаемых. Проблема состояла в том, что реализованных практик, подходящих для высшей школы на данный момент нет, поэтому для поднятия качества математической было решено провести данное исследование.

Главным аспектом применения при изучении высшей математике принципов уровневой дифференциации будем считать увеличение помощи студентам в образовательном процессе, а не увеличением содержания дисциплины.

Для успешного решения этих задач будут использованы дидактические приемы создания благоприятных условий для дифференцированного обучения:

- применение теоретических знаний к решению проблем преподавания математики,
- разнообразие и полнота использованного материала,
- преемственность и последовательность изложения учебного материала от простого к сложному, от знания к умениям и навыкам,
- формирование познавательной деятельности и творческой активности студентов,
- привитие студентам навыков самостоятельного усвоения учебного материала и умение применять их в практической работе,
- создание системы контроля и самоконтроля обучаемых,
- применение компьютерных технологий.

Дифференциация обучения поможет избавиться от необходимости учить всех одинаково, усилит гуманистическую направленность образования, позволит учитывать особенности, возможности и интересы студентов.

Дифференцированное обучение математике в технических вузах следует рассматривать как целостный процесс подготовки личности с учетом ее интересов, способностей, склонностей и возможностей, процесс, в котором студент и преподаватель выступают равноправными, заинтересованными и активными участниками. Отношение студентов существенно меняется.

Обучение становится более гибким и эмоциональным, а ситуация выбора вынуждает студента анализировать и оценивать свои собственные способности и возможности, уровень подготовки. Существенно облегчается работа со студентами, проявившими повышенные способности и интерес к определенной деятельности.

Для того чтобы правильно организовать процесс обучения в условиях дифференциации, уровни знаний студентов должны быть корректно определены. Существует множество особенностей, которые должны быть учтены, а именно познавательная деятельность, скорость обработки информации, замотивированность на успех, прилежность, дисциплина. Перед преподавателем стоит тяжелая задача донести до всех учебный материал, учитывая комплекс индивидуальных свойств каждого.

На основании исследования П.Г. Пичугиной [5], а также рассмотренных выше индивидуальных особенностей, влияющих на результат обучения выделим 3 уровня математической подготовки студентов (рисунок 3).

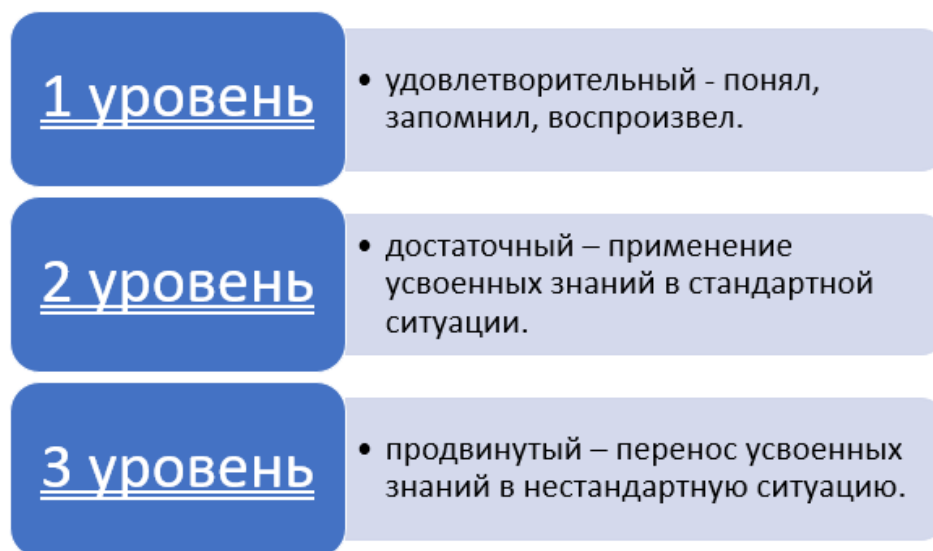


Рисунок 3 – Распределение уровней

Студент первого уровня демонстрирует базовые знания и понимание изучаемых понятий с недостаточным интересом к высшей математике, такие ученики часто пропускают занятия, недооценивая всю важность процесса обучения, им тяжело организовать самостоятельную работу [6], направленную

на отработку изучаемых понятий. Часто преподавателю необходимо больше времени уделять таким студентам, для того чтобы повысить их мотивацию, стимулировать их интерес к элементам высшей математики и как правило при освоении других дисциплин такие студенты испытывают трудности. Обучаемые, находящиеся на данном уровне способны формулировать простые выводы, анализируя информацию, представленную в виде графиков, таблиц и рисунков.

Ко второму уровню следует отнести студентов, которые на фоне достаточно невысокого уровня обучаемости все же могут достичь хорошего уровня в учебе. Это может быть достигнуто за счет прилежания, организованности, стремления использовать рациональные приемы организации своей деятельности. Такие студенты характеризуются нежеланием работать систематически, тщательность выполнения заданий, аккуратностью [7].

К третьему уровню следует отнести студентов, имеющих высокий уровень обучаемости, такие ученики проявляют глубокие знания, понимание терминов и понятий высшей математики. Их познания часто носят прикладной характер, у них стабильная дисциплина. Самостоятельная работа с литературой и дополнительными источниками информации не вызывает у них трудностей [7]. Часто студенты продвинутого уровня могут оказать помощь студентам, которые находятся на более низком уровне обученности. Такие студенты никогда не имеют долгов, все контрольные сдают вовремя, и как правило такие успехи демонстрируют при изучении других дисциплин. Они ответственны и добросовестны. Третий уровень самый желанный среди педагогов.

Процесс обучения должен быть построен таким образом, чтобы процентное соотношение таких студентов постоянно росло. Этого можно добиться как раз при внедрении различных технологий в образовательный процесс.

В исследовании предполагается строить дальнейшую работу на основе отраженных далее особенностей.

1.2 Компьютерное тестирование как средство определения уровня обученности студентов

Тестирование является наилучшей формой для проверки знаний по математике, по причине точности математических понятий, их четкой сформулированности; а также это связано с тем, что высшая математика относится к естественно-научному циклу, без четкой формулировки изучаемых понятий достижение при изучении данной дисциплины вряд ли возможно [8].

При использовании тестирования для определения уровня знаний при обучении математики задания будут подобраны разной степени сложности для того, чтобы при анализе полученных результатов был правильно определен уровень обученности тестируемых студентов, все это нужно для того, чтобы на основе сделанных выводов в дальнейшем корректировать учебный процесс [9].

При определении уровня знаний студентов будет проводиться тестирование, так как это самый удобный способ проверки знаний обучающихся. С помощью тестирования возможно определить, готов ли студент к изучению новой темы, в которой используются знания по ранее изученному материалу. Это возможно, так как при составлении теста вопросы и задания составляются таким способом, чтобы учитывать различные аспекты обученности студентов. Важным моментом тестирования является его компьютерная реализация, что существенно сокращает время проверки результатов тестирования.

Вообще говоря, тестирование – это исследовательский метод, который позволяет выявить уровень знаний, умений и навыков, способностей и других

качеств личности, а также их соответствие определенным нормам путем анализа способов выполнения испытуемым ряда специальных заданий.

Преимущества тестирования оценены и применяются достаточно широко, одной из причин является возможность при проведении теста определения уровня состояния развития некоторого свойства в объекте исследования и сравнение его с стандартом или с развитием этого показателя у исследуемого в более ранний период [10].

Для более эффективного проведения данного исследования, так как обучающихся достаточно много, была выбрана компьютерная форма тестирования. Достоинства компьютерной формы проведения тестирования и обработки результатов очевидны. Во-первых, при выборе неправильного ответа система может дать подсказку по теоретическому материалу, в котором студент допустил ошибку, во-вторых, при проведении компьютерного тестирования можно получить такую информацию о тестируемом как: количество допущенных ошибок выполненного задания, сколько времени студент потратил на ответ конкретного вопроса, в- третьих, задания в тесте можно варьировать по различной степени сложности [11].

В силу проводимого исследования рассмотрим такое понятие как критериально-ориентированный тест [12].

Проведение данного вида тестирования подходит для определения уровня подготовленности каждого тестируемого относительно требований учебной программы. Критериально-ориентированный тест позволяет соотнести учебные достижения отдельных студентов, чтобы определить, какую долю учебного материала они усвоили. На основе полученных результатов возможно разделения студентов на группы на основании заранее выбранного критерия. Выделим возможные три уровня: удовлетворительный, достаточный и продвинутой. Для отнесения к тому или ному уровню испытуемый должен достичь необходимого минимального для этой группы

уровня овладения учебным материалом. Удовлетворительный уровень устанавливается как норма, на основании чего студент в зависимости от прохождения процесса тестирования соотносится к рассмотренным ранее.

При разработке и составлении теста, реализуемого с помощью компьютерной системы, сначала была определена цель тестирования, а именно, определение уровня знаний студентов, при составлении заданий были использованы задачи различной степени сложности, после выполнения теста будет учитываться время выполнения каждого задания и наличие ошибок, тесты составляются по всем темам высшей математики, будет также отслеживаться повышается уровень студента либо снижается и что на это повлияло. Предполагается корректировка состава задач в зависимости от полученных результатов. Тесты предполагается использовать различных видов: с выбором ответа, открытые, закрытые и т.д.

Отметим все положительные аспекты компьютерного тестирования в отличии от тестирования на бумажных носителях. Проведение компьютерного тестирования позволяет повысить эффективность тестирования за счет уменьшения продолжительности теста [13]. Данный вид тестирования позволяет сэкономить траты на расходный материал, обработка результатов тестирования происходит автоматически, исключая ошибки, результат всегда объективен, и оценивание испытуемого можно увидеть сразу, с более подробной статистикой, так важной в проводимом исследовании.

Представленные преимущества компьютерного тестирования дали идею использовать данный вид контроля для определения уровня обученности студентов при проведении анализа данных на учебных занятиях по высшей математике в разрезе проводимого исследования.

После принятия решения о использовании тестирования как средства определения уровня обученности студентов высшей математике, был проведен анализ компьютерных систем, с помощью которых возможно

осуществить процесс тестирования. При выборе средства компьютерного средства для проведения тестирования, отдавалось предпочтение той системе, которая обладает максимальной оценочной способностью, использование которой позволит не только провести тестирования для того, чтобы проверить как тот или иной студент освоил учебный материал, но и так же определить время выполнения каждого задания, подсчитать количество допущенных ошибок и определить уровень знаний конкретного студента. Для эффективного контроля знаний, а также развивающегося обучающего тестирования, требуются механизмы, максимально уточняющие результат выполнения теста [14]. Для возможного использования компьютерной системы тестирования необходимо сформировать базу знаний, которую в дальнейшем используют для обнаружения и ликвидации проблемных мест при изучении того или иного раздела высшей математики. Как показывает практика, применения компьютерных технологий в традиционной системе обучения высшей математики повышает качество учебного процесса в высшем учебном заведении и усиливает мотивацию студентов в получении знаний [15].

Перечислим требования к системам компьютерного тестирования, которые важны при их выборе для реализации педагогического процесса (рисунок 4). Грамотно выделенные требования и их соблюдение значительно уменьшают время, которое будет потрачено для правильного определения уровня обученности студентов, что лежит в основе проводимого исследования, цель которого усовершенствовать процесс получения знаний студентами для их дальнейшего развития.

В силу проведенного анализа современных систем компьютерного тестирования рассмотрим более подробно такую систему электронного обучения как Moodle.

комплексность

- система должна обеспечивать весь цикл тестирования – разработка теста, предъявление его учащемуся, прием и оценивание его ответов, сбор и анализ статистической информации.

надежность в работе

- невосприимчивость к попыткам некорректного ввода ответов и другим несанкционированным действиям сдающего; сохранение промежуточных результатов (ответов) с возможностью возобновления процедуры после нештатного ее прерывания; устойчивость функционирования в локальной или глобальной сети.

Производительность

- возможность параллельного обслуживания многих сдающих, быстрое оценивание результата сдачи, оперативность передачи данных (заданий, результатов) в сетевых режимах использования, малое время отклика серверных приложений, малое время обработки конечных результатов.

Масштабируемость

- возможности использования СКТ на отдельных компьютерах, в локальных сетях и, возможно, в глобальной сети.

Защищенность

- обеспечение невозможности доступа сдающих к эталонным ответам, учащихся и посторонних лиц – к общим результатам тестирования.

Мультимедийность

- возможность встраивания в тестовые задания мультимедийных объектов, а также разнообразие анализаторов ответов сдающего (клавиатура, мышь).

полнота типов тестовых заданий

- обязательная поддержка пяти основных типов тестовых заданий (открытого типа и закрытого: одиночный и множественный выбор, соответствие, установление последовательности).

простота эксплуатации

- стандартный современный интерфейс, четкое и эффективное разделение на функциональные подсистемы, ориентация на стандарты в информационной индустрии, единый стиль оформления программ, документация для каждой группы пользователей, прозрачный смысл настраиваемых параметров, наглядная и удобная для педагога форма представления итогов тестирования (включая оценку качества теста).

Рисунок 4 - Требования к системам компьютерного тестирования Moodle (Modular Object-Oriented Dynamic Learning Environment) – популярная бесплатная система электронного обучения. Область применения, как правило, дистанционное обучение студентов [17]. У системы есть плагины

и модули, благодаря которым можно изменить дизайн обучающей среды и расширить возможности. Использовать можно бесплатную серверную версию, в том случае, если программа была установлена на сервере.

Данная система позволяет создавать онлайн-курс, с презентациями, видео и текстами. Курс можно сделать доступным отдельной группе людей или всем. Под курсами можно оставлять комментарии, обсуждать тему урока или задавать вопросы преподавателями. Также для общения можно создавать беседы на форуме. Возможна разработка базы знаний. В базу знаний можно добавить учебные материалы, которые доступны всем пользователям в любое время. Так же система автоматически следит за успеваемостью студентов и составляет отчеты. Например, там можно посмотреть, какие ошибки чаще всего допускают студенты. В Moodle есть редактор текстов с пятнадцатью типами заданий. Например, можно выбирать один правильный объект или перетаскивать объекты. Ещё можно ограничить время выполнения заданий и число попыток. Moodle может автоматически проверять ответы, показывать ошибки.

Остановимся подробнее на тестировании, которое можно проводить в системе Moodle. Во-первых, все используемые вопросы сохраняются в банке вопросов, благодаря чему далее можно создавать тесты как по отдельным темам курса, так и по всему курсу целиком, используя задачи и задачи разной степени сложности. Во-вторых, система поддерживает использование графических иллюстраций в тесте, что очень удобно. В-третьих, вопросы и задачи можно распределить и сгруппировать по системе категорий/подкатегорий в соответствии с их принадлежностью к конкретным темам, разделам и подразделам курса, а на самом нижнем уровне к группам вопросов, однородных по сложности и тематике. Банк вопросов учебного курса должен содержать, по крайней мере, несколько сотен тестовых заданий.

Для проведения каждого конкретного тестирования, например, для определения уровня усвоения той или иной темы, в курсе Moodle создается отдельный тест, параметры которого настраиваются разработчиком тестов, и

который затем наполняется конкретными заданиями из Банка вопросов в соответствии с целями тестирования. Настройка параметров теста и формирование его состава осуществляется в соответствии с целями тестирования, особенностями его проведения, контингентом тестируемых студентов и др. Например, определяются временные рамки предъявления теста студентам, форма выдачи студентам вопросов теста и т. д. Этот процесс называют еще формированием сценария теста.

Таким образом компьютерная тестовая система Moodle, позволяет сформировать для студентов набор разнообразных тестов, которые будут использованы для определения уровня обученности и усвоения той или иной темы. Каждый тест будет являться в курсе самостоятельным элементом оценивания. Оценки, получаемые студентами за выполнение тестов, используются преподавателем в качестве характеристик успешности освоения студентами учебного материала наряду с оценками, получаемыми за работу с другими элементами оценивания.

1.3 Анализ возможностей использования нейронных сетей при обучении математике

Многолетние наблюдения позволили сделать тот вывод, что в настоящее время существует большой разрыв между стремительно развивающимися техническими науками и способами получения знаний студентами, обучающихся в технических вузах. Для максимального их сближения считаем целесообразным внедрение в образовательный процесс всевозможных компьютерных технических средств, среди которых лидирующие места занимают искусственные нейронные сети [20].

На сегодняшний момент использование искусственного интеллекта носит прикладной характер. Нейронные сети и системы применяются для решения достаточно широкого спектра задач, в том числе и математических, где требуется найти какое-либо значение на основе известных данных проведенных исследований, если этих данных очень много. Искусственный интеллект нашел свое применение в робототехнике, при обеспечении

безопасности, поисковых системах, математическом анализе, юриспруденции, медицине, образовании и многих других направлениях.

Рассмотрим некоторые возможности использования искусственного интеллекта в образовательном процессе, в частности при изучении высшей математики. Во-первых, при проведении математических вычислений можно проводить упрощение математических выражений, можно использовать всплывающие подсказки при решении различных видов уравнений и неравенств.

Во-вторых, можно использовать самое распространенное свойство нейронных сетей при распознавании образов, в математике данную возможность можно применять при составлении портрета обучаемого, для построения модели поведения студента в той или иной ситуации, для мониторинга динамики роста или наоборот снижения тех иных признаков (умственной деятельности, заинтересованности в предмете в зависимости от каких либо факторов и условий), для того чтобы иметь возможность управления и корректировки процесса обучения, а так же для того чтобы была возможность классифицировать учеников по тем или иным признакам [21].

В-третьих, принципы действия нейросетей можно применять при прогнозировании успеваемости студентов во время обучения высшей математики. В процессе прогнозирования подаются значения переменных (успеваемость студента, пропуски), эти значения в зависимости от типа прогнозирования обрабатываются и предсказываются. Чем больше будет данных, представленных для проведения процесса прогнозирования, тем точнее будет результат.

Прогнозирование результатов учебного процесса, с целью определения уровня обученности, является одной из наиболее актуальных задач, с которыми сталкиваются высшие учебные заведения. Предсказание успеха студентов может помочь университетам определить тех студентов, которые нуждаются в дополнительной поддержке, и предоставить им индивидуальный подход к обучению. Нейронные сети являются мощным инструментом для

анализа и обработки данных, что позволяет прогнозировать результаты учебного процесса на основе различных факторов, таких как история успеха студента, методы обучения и даже личные характеристики студента [22]. Исследование посвящено возможности использования данных о уровне усвоения той или иной темы студентом для прогнозирования его успеха в учебе.

Иногда, при изучении некоторых разделов высшей математики, наиболее важно предугадать момент, когда текущий уровень успеваемости студента изменит свое направление на более низкий или высокий, что входит в рамки данной исследовательской работы, поэтому не так важно значение результата прогнозирования, а именно какой уровень обученности студента будет, как ответ на вопрос, когда студент повысит или, наоборот, снизит свой уровень обученности и какие факторы могут на это повлиять. Обучающая выборка и точность прогноза, требуемая для конкретной проблемы, имеют большое значение на прогнозирующую систему.

Использование нейросетей может помочь усовершенствовать учебный процесс таким образом, что в значительной мере помочь облегчить работу преподавателей, а именно, с помощью некоторых возможностей искусственного интеллекта можно создавать презентации, автоматически проверять выполненные задания, при преподавании высшей математики можно использовать интересные идеи для проведения занятий, то есть нейронные сети могут подстраиваться под индивидуальные потребности обучающихся, что входит в пределы проводимого анализа. Достаточно хорошо обученная нейронная сеть может найти свое применение, как средство автоматизированного создания и проверки учебных заданий и тестов, что в значительной мере сокращает время для подготовки к занятиям педагогического состава, освобождая больше времени для проведения научной работы. Нейронные сети можно использовать как средство для подготовки и оценки заданий, и предоставление обучающимся своевременную помощь и реальное состояние своей успеваемости [22].

С помощью нейронных сетей можно создавать учебные материалы по высшей математике, которые будут составлены таким образом чтобы при выполнении студентом различных заданий учитывался их уровень знаний, их способность обрабатывать информацию, что реализует один из принципов образования, как индивидуализация.

Кроме того, нейронные сети можно использовать при создании интерактивных учебных материалов, которые могут представлять математические понятия в наглядной и понятной форме, что позволит более эффективно обеспечить студентов всевозможными математическими знаниями, умениями и навыками при изучении математических понятий. Это в большей мере позволит помочь студентам визуализировать и усвоить сложные математические идеи, давая им возможность более глубоко понять их суть.

Еще одним направлением использования возможностей искусственного интеллекта можно отметить создание индивидуальной образовательной программы, которая могла бы облегчить контроль и анализ обучения высшей математике, например, классифицировать и группировать студентов, учащихся по возрасту, уровню знаний, способностям или наклонностям.

Внедрение компьютерных технологий в образовательный процесс, в частности, некоторых свойств нейронных сетей, помогут разнообразить процесс обучения высшей математике, а именно, сделать его более интересным и содержательным, например, решить проблему зубрежки, или же повысить мотивацию в приобретении знаний при изучении нового материала. Обучение приносит больше результатов, если в нем много практических заданий и графической интерпретации некоторых понятий, благодаря которым лучше закрепляются полученные знания на практике.

Использование нейронных сетей в обучении возможно при проверке знаний с помощью тестирования [23], например, в целях определения уровня обученности. На основе входных данных, в нашем случае – полученные данные при проведении тестирования, в частности это время выполнения того

или иного задания и количество допущенных ошибок, будут получены выходные данные, анализ которых даст точную информацию об уровне обученности студента, с целью дальнейшего использования данной информации в образовательном процессе. Искусственная нейронная сеть сразу выдаст комментарии студенту, которые помогут устранить пробелы в знаниях. Преподаватели лишь контролируют процесс, создают интерактивные курсы по различным предметам [24].

Использование искусственного интеллекта после проведенного тестирования может помочь провести анализ полученных данных, на основе которого преподаватель может проанализировать изучение каких разделов высшей математики вызывает у обучающихся наибольшие трудности, чтобы при дальнейшей объяснении данного курса или темы уделить больше времени и внимания на нее. Но самое главное помнить, что все должно быть в меру, так как компьютерные технологии должны использоваться в образовательном процессе как помощь преподавателю на любом этапе обучения, а не заменить его, «живое» общение, возможности нейронных сетей должны лишь облегчить рутинную работу преподавателя.

Рассмотрев положительные аспекты использования нейронных сетей в процессе обучения математики, стоит отметить и негативные моменты. Как правило, при реализации самостоятельной работы, а также при дистанционной форме обучения, как показала практика на карантине 2020-2022 гг. во время пандемии, достаточное количество студентов испытывали трудности с самостоятельной работой, особенно в онлайн-курсах, где нет непосредственной поддержки преподавателя. Кроме того, некоторые студенты могут испытывать затруднения с использованием технологий или предпочитать традиционные методы обучения [25]. Для преодоления этих негативных моментов важно разнообразить методы преподавания, чтобы каждый студент мог найти подходящий для себя. Это может включать в себя комбинацию традиционных лекций, групповой работы и использование современных технологий.

Все большую актуальность приобретает внедрение инклюзивного образования, для обучения людей с ограниченными возможностями применение компьютерных технологий намного облегчит процесс получения необходимых знаний. Для студентов, страдающих аутизмом, при использовании искусственного интеллекта в обучении наблюдается значительное улучшение фонологического осознания и навыков чтения. Обучаемые со значительными нарушениями речи и ограниченными физическими возможностями также выигрывают от применения возможностей нейронных сетей в учебном процессе, обладающих достаточной гибкостью, что позволяет подстраиваться под индивидуальные потребности. Для слабослышащих и глухих визуальное представление информации значительно повышает мотивацию к учебе.

У студентов без отклонений и обучаемых с физическими особенностями применение в традиционном обучении компьютерных технологий повышает мотивацию к приобретению знаний, студенты в режиме реального времени могут отследить свои достижения при изучении элементов высшей математики. Повышается мотивация, дисциплина, желание узнать и понять как можно больше новой информации, которую можно применить в нестандартных ситуациях, применить теорию на практике без боязни допустить какие-либо ошибки [25].

Проведенный обзор основного инструментария нейронных сетей, который можно использовать при изучении элементов высшей математики с целью повышения эффективности образовательного процесса, показал, что после появления подобных нововведений в учебный процесс во многом повышается его качество и реализация основных направлений учебного процесса.

В рамках проведенного исследования выделим основные направления, которые необходимо устранить из процесса обучения для того, чтобы он стал намного эффективнее.

Во время практических занятий по высшей математике и при самостоятельной работе, которая часто организуется обучаемыми не со всем корректно, в силу неправильного распределения свободного времени после учебы, у студентов не хватает времени, чтобы проанализировать полученную информацию и в должном объеме закрепить полученные умения и навыки. Именно внедрение нейронных сетей, использование их возможностей позволит уменьшить эти пробелы, так как появится возможность модернизировать и усовершенствовать педагогический процесс путем использования созданных эффективных программ обучения, которые могут сократить время, необходимое на обучение и правильно выстроит его структуру, особенно на начальном этапе обучения. Это повысит эффективность обучения и мотивацию студентов на получение и закрепление новых знаний. Применение нейросетей в образовании можно использовать для создания индивидуальных программ обучения, с учетом потребностей и уровня знаний студентов. Такая программа может конкретно для каждого студента оптимизировать время обучения, а также предложить наиболее простой или сложный материал, исходя из уровня знаний студента.

Использование искусственного интеллекта позволит поднять уровень познавательной активности обучающихся при групповых занятиях, так как будет решена проблема индивидуализации обучения, каждый студент сможет получать индивидуальную программу обучения, учитывающую его уровень знаний, скорость обучения, профессиональные интересы и прочее.

Преподавателю будет намного проще организовывать свою методическую работу, если некоторые его действия будут выполнять успешно обученные нейронные сети. В частности, разработанная нейросеть позволит правильно определить уровень знаний студентов после обработки результатов тестирования, что позволит вовремя обнаружить у студентов снижение активной мыслительной деятельности. Рассмотрим основные направления применения нейронных сетей в процессе обучения (таблица 1).

Таблица 1 - Основные направления применений нейронных сетей в процессе обучения

Основные направления применений нейронных сетей в процессе обучения		
проведение практических занятий по математике и, в частности, для самостоятельного эвристического поиска доказательств студентами	осуществление контроля знаний	индивидуальная работа обучающихся и самоподготовка

В рамках проводимого исследования будет создана нейросеть, которая, обрабатывая информацию об успеваемости конкретного студента, а именно, количество минут, которое студент потратил на выполнение того иного задания, количество ошибок, которое он допустил, будет определять уровень обученности студента на основе входных данных [27]. В зависимости от ситуации и этапа педагогического процесса правильно обученная нейросеть может предложить студенту, испытывающему трудности при изучении, конкретную помощь в виде материала, который может быть представлен в виде графиков или других иллюстраций, так как иногда графическая информация более информативна, возможно нейросетью будет предложена литература, практические задания, выполнив которые студент сможет повысить свой уровень.

Целью исследования можно считать создание нейронной сети, обучение и внедрение которой в учебный процесс приведет к повышению интеллектуального и профессионального уровня специалистов.

2 Разработка нейросетевой модели для определения уровня обученности студента при изучении высшей математики

Постановка задачи

Задача состоит в определении уровня знаний студентов при изучении высшей математики с целью повышения эффективности процесса обучения в вузе с помощью использования нейронных сетей для оценивания дальнейшей динамики изменений роста познавательной деятельности студентов.

Будет оцениваться время выполнения студентами задач различной степени сложности и фиксирование ошибок, допущенных обучаемыми на каждом этапе.

2.1 Набор данных для обучения нейронной сети

Одной из основных задач высшего учебного заведения является предоставление студентам качественных знаний, которые отвечали бы потребностям современного рынка труда. Реализация данной задачи возможна с помощью разработки образовательных программ с учетом новых информационно-коммуникационных технологий.

Основываясь на научно-исследовательскую работу, проведенную за время обучения в магистратуре, перечислим целесообразность использования нейронных сетей при изучении высшей математики в вузе на примере изучения темы «Дифференциальные уравнения»:

- исследование некоторых математических моделей, например, при решении дифференциальных уравнений второго порядка, важны для формирования «нелинейного» мышления инженера,
- решение дифференциальных уравнений включает в себя не только знание методов решения дифференциальных уравнений, но и знание методов интегрирования,
- недостаточное количество времени для изучения всех видов дифференциальных уравнений, которые имеют широкое прикладное значение,

– студенту необходимо видеть связь между лекционным и практическим курсом изучаемой дисциплины, и своей будущей профессией. Это повышает мотивацию к учебе и развивает творческое мышление,

– решение дифференциальных уравнений развивает умение исследовать математическую модель, что предоставляет изучать явление в целом, предсказывать его развитие, делать количественные оценки измерений.

Для практической реализации проводимого исследования будет создана система, которая после тестирования, проводимого с целью выявления уровня обученности студента делила бы группу на подгруппы различного уровня обученности, в зависимости от результатов входного тестирования. Далее после проведения методических мероприятий, направленных на повышение познавательной деятельности студентов, будет проводиться тестирование для того, чтобы увидеть положительную динамику в развитии студентов.

Возможные этапы использования нейронных сетей при изучении какого-либо раздела высшей математики:

– тестирование, определение уровня математических способностей студента,

– при возникновении проблем с решением, нейронная сеть выдает комментарии для устранения пробелов в знаниях,

– после определения уровня студенту предлагается набор заданий для отработки данной темы,

– при желании студента повысить уровень изучаемой темы, ему предлагается задания на уровень выше, также с появляющимися комментариями для устранения пробелов в знаниях.

Приведем пример набора заданий разного уровня при изучении темы «Дифференциальные уравнения с разделяющимися переменными [28, 29, 30]».

Первый уровень.

Задание: решить дифференциальное уравнение, найти общее решение.

$$x \cdot y' - y = 0, \quad x \cdot y' + y = 0, \quad y \cdot y' + x = 0, \quad y' = y, \quad x^2 \cdot y' + y = 0.$$

Задание: решить дифференциальное уравнение, найти общее и частное решение, удовлетворяющее начальным условиям.

$$2y' \cdot \sqrt{x} = y, y(4) = 1, y' = (2y + 1) \cdot \operatorname{ctgx}, y\left(\frac{\pi}{4}\right) = \frac{1}{2}$$

$$x^2 y' + y^2 = 0, y(-1) = 1, (1 + e^x)yy' = e^x, y(0) = 1$$

$$y' = 2\sqrt{y} \cdot \ln x, y(e) = 1.$$

Второй уровень.

Задание: решить дифференциальное уравнение, найти общее решение.

$$y' + \sqrt{\frac{1-y^2}{1-x^2}} = 0, \sqrt{1-y^2}dx + y\sqrt{1-x^2}dy = 0, e^{-s} \cdot \left(1 + \frac{ds}{dt}\right) = 1$$

$$y' = 10^{x+y}, y' + \sin \frac{x+y}{2} = \sin \frac{x-y}{2}.$$

Задание: решить дифференциальное уравнение, найти общее и частное решение, удовлетворяющее начальным условиям.

$$y' = \frac{1+y^2}{1+x^2}, y(0) = 1, \sin y \cdot \cos x dy = \cos y \cdot \sin x dx, y(0) = \frac{\pi}{4}$$

$$y - xy' = b \cdot (1 + x^2 \cdot y'), y(1) = 1,$$

$$x^2(y^2 + 5) + y^2(x^2 + 5) \cdot y' = 0, y\left(\frac{\pi}{2}\right) = 0,$$

$$\frac{dy}{dx} + (2y + 1) \cdot \operatorname{ctgx} = 0, y\left(\frac{\pi}{2}\right) = 0.$$

Третий уровень.

Задание: решить дифференциальное уравнение, найти общее решение.

– вывести зависимость $U(t)$ на обкладках конденсатора при его разрядке в RC-цепи,

– найти массу радиоактивного вещества спустя время равное четырем периодам полураспада, если начальная масса составляет 64 г,

– найти линию, проходящую через точку (2,3) и обладающую свойством: отрезок любой ее касательной, заключенной между координатными осями, делится пополам в точке касания,

– материальная точка массой 1 гр. движется прямолинейно под действием силы, прямо пропорциональной времени, отсчитываемому от момента $t = 0$ и обратно-пропорциональной скорости движения точки. В

момент $t = 10$ секунд скорость равнялась 50 см/сек, а сила 4 динам. Какова будет скорость спустя минуту после начала движения?

– какой поверхностью вращения является зеркало прожектора, если лучи света, исходящие из точечного источника отразившись, направляются параллельным пучком?

После проведенного тестирования были получены и обработаны данные студентов, представленных в таблице 2, чтобы на их основе создать обучающую и тестируемую выборки для обучения нейронной сети.

Таблица 2 – Распределение уровней обученности

Уровень обученности	Время решения задачи, мин	Количество ошибок
удовлетворительный light	Сложный уровень 19-21 Средний уровень 14-18 Низкий уровень 10-12	0-2
достаточный middle	Сложный уровень 10-15 Средний уровень 8-10 Низкий уровень 5-7	0-2
продвинутый hard	Сложный уровень 7-10 Средний уровень 3-5 Низкий уровень 1-3	0-2

После того, как данные будут загружены, их необходимо предварительно обработать. Это включает нормализацию данных и преобразование их в допустимый формат.

Нормализация является важным этапом предварительной обработки данных. Он включает в себя масштабирование данных таким образом, чтобы они имели среднее значение 0 и стандартное отклонение, равное 1. Это помогает гарантировать, что модель сможет более эффективно обучаться на основе данных.

После предварительной обработки данных их нужно разбить на обучающий и тестовый наборы. Обучающий набор используется для обучения модели, а тестовый набор — для оценки производительности модели [32].

После подготовки данных можно приступить к созданию модели. Это включает в себя определение сетевой архитектуры и обучение модели.

2.2 Выбор архитектуры нейронной сети

Основная идея искусственных нейронных сетей в имитации работы естественных нейронных сетей. Поэтому суть искусственных нейронных сетей можно выделить, рассмотрев приближённую работу человеческого мозга, который является естественной нейронной сетью. Работа естественной сети состоит из взаимодействия огромного количества биологических нейронов (рисунок 5) – клеток, основные составляющие которой: ядро, отростки, тела и компоненты для связи с другими нейронами. Через связь нейронов друг с другом передаются электрохимические импульсы, приводящие всю сеть в состояние спокойствия или возбуждения. В итоге мозг принимает сигнал от раздражителя, обрабатывает его и решает, как действовать в сложившейся ситуации. Действие может быть рефлекторным или результатом сложного процесса рефлексии. В принятии решения играет роль опыт, который хранится в архивах памяти.

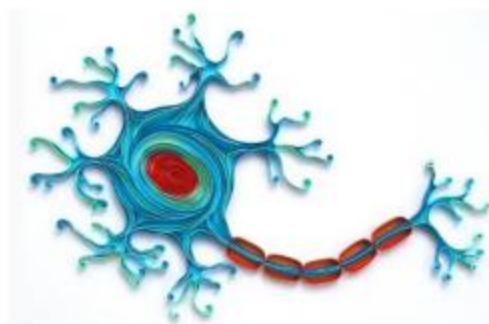


Рисунок 5 – Биологический нейрон

По упрощенной теории естественной нейросети проведём изучение структуры биологических нейронов и принципах их работы. Основу нейрона составляют тело клетки и ядро. У тела существует огромное количество ответвлений, которые называются дендриты. Среди дендритов можно выделить особо длинные, которые принято называть отдельным словом «аксоны». Они используются для связи между нейронами, которая реализуется с помощью синапсов (места контакта двух нейронов, по которым проходит импульс). На дендрит приходит сигнал, который по аксону

передается другому нейрону. В точке синаптической передачи импульса, его частота и амплитуда может изменяться. Это общая схема работы человеческого мозга. В 1957 году психологом Корнеллского университета Франком Розенблаттом была продемонстрирована первая обучаемая искусственная нейронная сеть, которая называлась «Перцептрон», и базировалась на принципах работы естественных нейронных сетей, описанных выше. Дендрит выступает роли информационного входа, аксон – выхода, тело нейрона послужило прототипом сумматора, а кодом, состоящим из 0 и 1 происходит регулирование частоты и импульса сигнала перед подачей на сумматор в разделе «Веса». Это основные составляющие искусственного нейрона (рисунок 6).

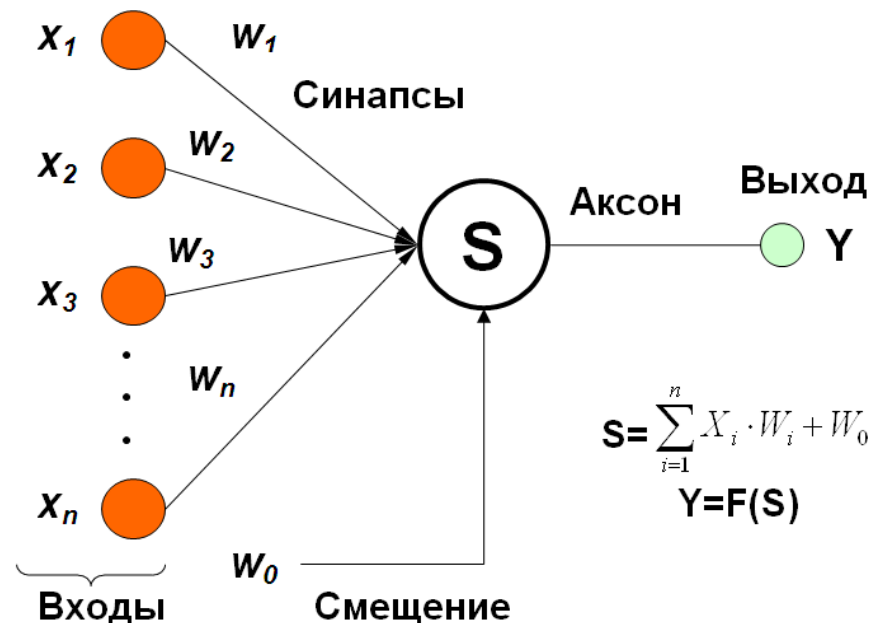


Рисунок 6 -Простейшая модель искусственного нейрона

Соединенные в достаточно большую сеть с управляемым взаимодействием, такие, по отдельности простые, процессоры вместе способны выполнять довольно сложные задачи.

Рассмотрим функцию активации

$$S = \sum_{i=1}^n X_i \cdot W_i + W_0,$$

где X_i множество входов, которые принимает нейрон – внешние данные,

W_i вес – внутренние параметры, т.е. коэффициент, на который умножается вход,

число W_0 - смещение – внутренние параметры.

В результате получаем нелинейную функцию активации (нелинейной она становится благодаря смещению), которая является внутренним параметром $F()$, она передает свое значение на выход Y .

Изменяя параметры нейрона, изменяется и его поведение, следовательно, подбирая параметры нейрона его можно обучать.

Учитывая всё вышесказанное, можно заключить, что искусственная нейронная сеть – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [33]. Вообще говоря, искусственная нейронная сеть представляет собой множество соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие, по отдельности простые, процессоры вместе способны выполнять довольно сложные задачи.

Для увеличения скорости работы программы, для более быстрого решения задачи, создают несколько нейронных слоев.

Входной слой. Система получает данные и раскладывает их на элементы. Так, изображения разбиваются на пиксели, и каждый из них отправляется на собственный нейрон.

Скрыты слои. Алгоритмы обрабатывают данные: система распознает их и интерпретирует. По отдельности нейроны могут решить только однотипные задачи - определить истинное и ложное утверждение. Но сигнал активирует тематически связанные нейроны - получают разветвленные алгоритмы. Когда нейросеть проанализировала все данные и определила все связи, она выдает решение.

Выходной слой. Суммирует итоговые данные и выводит результат.

Входные данные умножаются на веса и затем подаются в следующий скрытый слой. Смещение также задается в качестве входных данных наряду с взвешенными входными данными. Взвешенная сумма передается через нелинейную функцию, называемую функцией активации.

Среди основных топологий нейронных сетей можно выделить полносвязные, сверточные и рекуррентные нейронные сети.

Сверточная нейронная сеть – это тип нейронной сети, который используется для анализа и обработки изображений, видео, звука и других типов сигналов. Она была разработана для эффективной обработки больших объемов данных и получила широкое применение в таких областях, как компьютерное зрение, распознавание речи, обработка естественного языка и многих других [34].

Рекуррентная нейронная сеть – это тип нейронной сети, который используется для анализа последовательностей данных, таких как текст, звук, временные ряды и другие. Она была разработана для обработки данных, которые имеют зависимости во времени, что делает ее особенно полезной в таких областях, как распознавание речи, машинный перевод, генерация текста и многих других [35].

Многослойный персептрон – это одна из самых популярных моделей нейронных сетей, которая используется для решения широкого круга задач, включая классификацию, регрессию и прогнозирование. В этой модели нейроны организованы в несколько слоев, каждый из которых имеет свои входы и выходы [36]. Основным преимуществом данной модели является способность к обучению на больших объемах данных с высокой точностью [35]. Многослойный персептрон хорошо работает с данными, которые имеют множество числовых признаков.

Многослойные нейронные сети имеют несколько слоев, которые связаны между собой таким образом, что каждый нейрон последующего слоя имеет связь со всеми нейронами предыдущего слоя. Сложность сети резко

возрастает от увеличения размерности входных данных и от количества скрытых слоев. Другая проблема заключается в том, что в полносвязной сети изображения представляют собой одномерные последовательности и при этом не учитываются особенности изображений как структуры данных. Тем не менее, для изображений небольших форматов можно использовать и полносвязную сеть.

Будем использовать многослойные нейронные сети (рисунок 7) прямого распространения, так как они больше всего подходят для решения задач, связанных с анализом табличных данных.

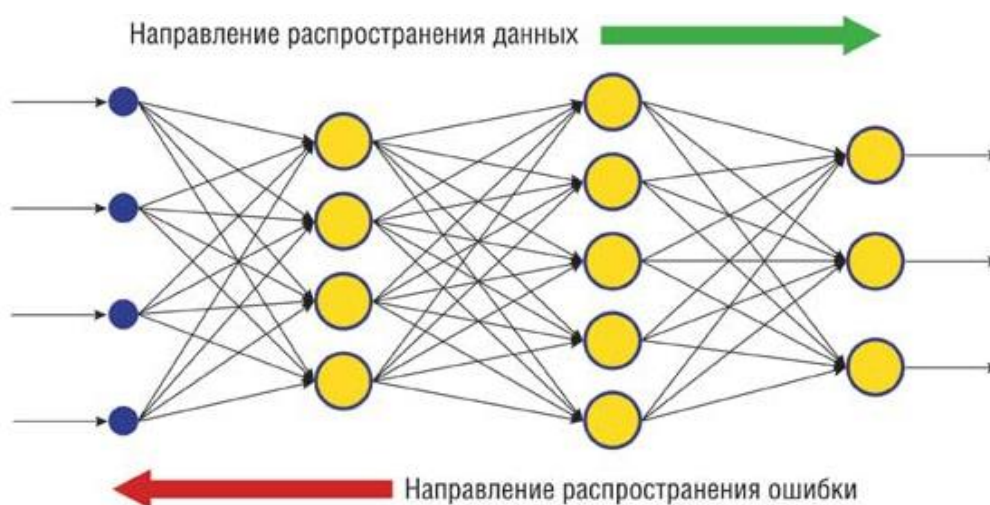


Рисунок 7 – Многослойный персептрон

Рассмотрим параметры персептрона. Существуют параметры (обучаемые числа) и гиперпараметры (необучаемые, но задаваемые числа). К гиперпараметрам можно отнести: число слоев нейронов и число нейронов в каждом слое.

Параметры персептрона - это параметры всех его нейронов. Каждый нейрон в персептроне имеет свои собственные параметры:

- веса, смещение,
- функция активации (но часто ее относят к гиперпараметрам, не обучают и выбирают одинаковой у всех нейронов слоя).

Для успешной работы нейронной сети её необходимо обучить, т.е. правильно настроить веса и смещения внутри сети. В этом процессе как раз и

применяются методы машинного обучения практически всех категорий, что обеспечивает глубокую связь нейронных сетей и машинного обучения.

Значительную роль при обучении нейронной сети играет правильно выбранная функция активации. Рассмотрим существующие функции активации.

Линейная функция представляет собой прямую линию и пропорциональна входу (то есть взвешенной сумме на этом нейроне).

Такой выбор активационной функции позволяет получать спектр значений, а не только бинарный ответ. Можно соединить несколько нейронов вместе и, если более одного нейрона активировано, решение принимается на основе применения операции $\max$ (или softmax).

Рассмотрим линейную функцию

$$f(x) = c \cdot x,$$

где c - коэффициент пропорциональности.

График кусочно-линейной функции

$$f(x) = \begin{cases} -1, & x \leq -0.5, \\ cx, & -0.5 < x \leq 0.5, \\ 1, & x > 0.5. \end{cases}$$

при $c = 2$ представлен на рисунке 8.

Для обучения нейронной сети используется метод обратного распространения ошибки, который подразумевает под собой использование производной для функции активации, в данном случае ее производная будет равна коэффициенту пропорциональности C . Нейронная сеть является математическим аппаратом, следовательно, ее описание можно описать математическими операциями, а в случае с линейной функцией многослойный перцептрон становится эквивалентным однослойному.

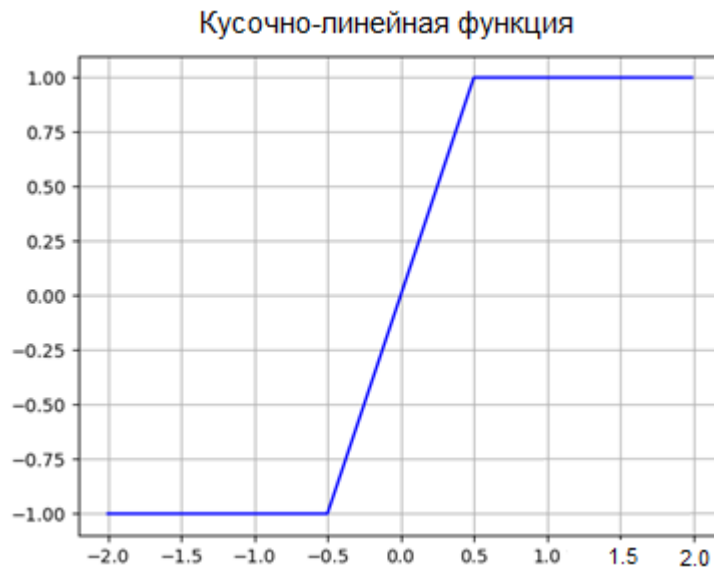


Рисунок 8 – Линейная функция

Функция ReLU (линейный выпрямитель) имеет вид:

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

График нелинейной функции ReLU представлен на рисунке 9.

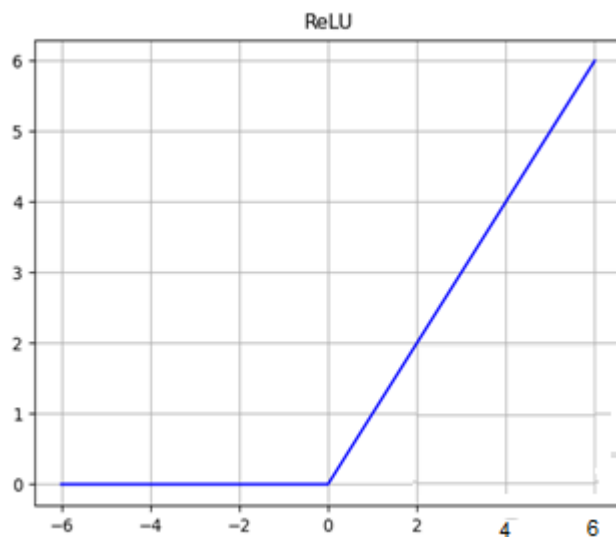


Рисунок 9 - Функция ReLU

Преимущество ReLU - это ее математический вид, функция требует меньше вычислительных ресурсов, что ускоряет физическое время обучения нейронной сети, это хорошо подходит для создания глубоких нейронных сетей. Использование ReLU позволяет инактивировать некоторые нейроны,

это опять же сделает нейросеть более быстрой в пересчете времени на одну эпоху обучения.

Но, как и у любой функции. у ReLU существуют недостатки, это связано с тем, что отрицательные входные значения дадут на выходе нулевой градиент, что негативно скажется на процессе обучения нейронной сети из-за отсутствия обновления весовых коэффициентов. Эта проблема частично решается модификацией ReLU, в которой при отрицательных значениях входного параметра присваивается некая линейная функция с маленьким коэффициентом пропорциональности, что позволяет избавиться от нулевого градиента и в процессе обучения корректировать веса.

Сигмоидная логистическая функция наиболее популярная активационная функция имеет вид:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Сигмоида очень часто применяется в задачах классификации, поскольку она является гладкой функцией и меняется в диапазоне от -1 до 1.

Еще одним достоинством логистической функции является вид ее производной. При вычислении производной используются те значения, которые были получены на предыдущем шаге, что экономит время за счет упрощения математических операций:

$$f'(x) = f(x) \cdot (1 - f(x))$$

Как видно на рисунке 10, производная имеет наибольшее значение в промежутке $[-2; 2]$, что позволяет входным значениям быстрее прижиматься к границам диапазона значений, иными словами, быстрее определять, насколько активен нейрон. Однако, приближение значений функции активации к

границам, опять же, приводит к затуханию градиента, и как следствие, почти не обновляемым весовым коэффициентам.

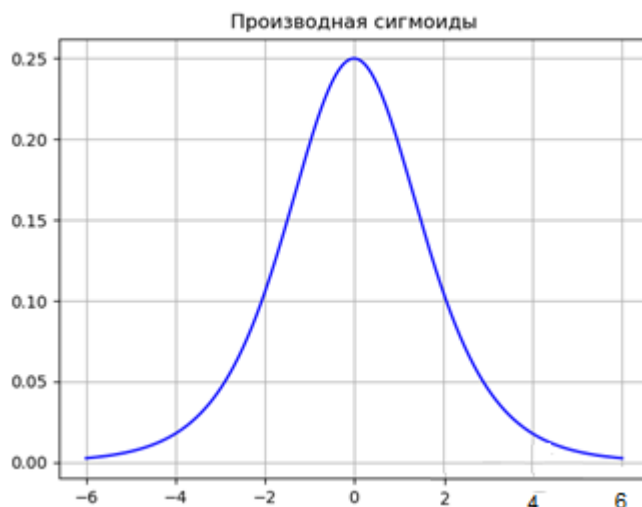


Рисунок 10 – График производной сигмоиды

Гиперболический тангенс представляет собой модифицированную сигмоиду:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1,$$

следовательно, все преимущества и недостатки сигмоиды в равной степени относятся и к тангенсу. Различия проявляются лишь в градиенте тангенциальной функции, он выше, это видно из сравнения производных, представленных на рисунке 11.

Выбор активационной функции.

Помимо формул, представленных выше, существуют множество других функций активации, исследователи в области искусственного интеллекта постоянно ищут новые способы улучшить нейронные сети, в том числе модифицируя уже существующие способы и приемы обработки входных сигналов. Выбор активационной функции одна из первоочередных задач для построения правильной нейронной сети, однако, не существует единого правила, в каком случае какую из функций применять. Есть лишь практика, которая показывает, что ReLU лучше показывает себя как аппроксиматор, а

сигмоида и гиперболический тангенс чаще используются в задачах классификации.

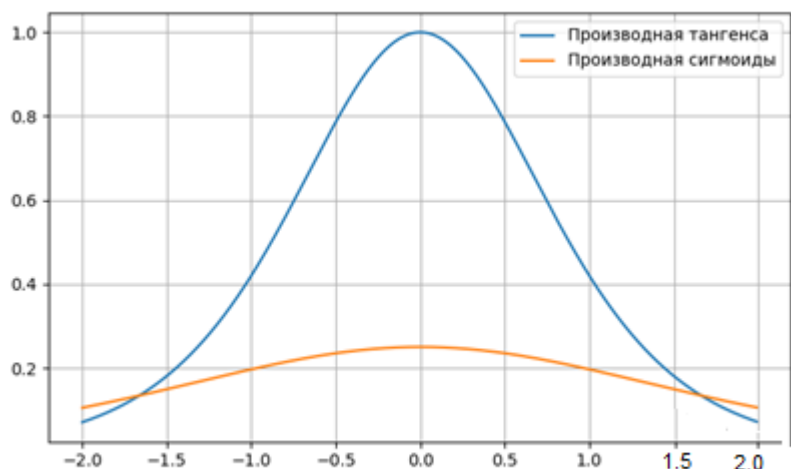


Рисунок 11 – Графики производных гиперболического тангенса и сигмоиды

При решении сложных задач с использованием многослойных сетей возможно комбинировать функции на разных слоях, чтобы добиться наиболее высокой точности и скорости обучения.

2.3 Описание разработанной НС

Обучение нейронных сетей представляет собой сложный процесс обработки данных, в результате которого сеть приобретает способность обрабатывать и распознавать данные. Данная процедура включает в себя последовательное представление данных на вход нейронной сети и сравнение выходных данных с их истинным значением, после чего вносится коррекция весовых коэффициентов нейронов в сторону уменьшения ошибки выходных данных. Этот процесс производится многократно с использованием определенного набора данных. В процессе обучения используется часть данных, которая носит название тренировочный набор. Один цикл обучения с использованием всего набора данных носит название эпоха. Как правило, для качественного обучения сети требуется много эпох. Процесс обучения нейронных сетей, имеющих много скрытых слоев, часто носит название глубокого обучения. В процессе обучения и контроля за качеством обучения широко используются методы градиентного спуска и обратного распространения ошибки [36]. Метод стохастического градиентного спуска

(рисунок 12) позволяет найти локальный минимум функции (которая является функцией ошибки) от нескольких переменных (весов нейронов) путем последовательного малого изменения этих переменных.

Рассмотрим более подробно данный метод оптимизации весов нейронной сети путем выполнения небольших шагов в направлении градиента функции ошибки.

Простыми словами, метод стохастического градиентного спуска похож на студента, который пытается выучить предмет, решая практические задачи. Студент решает по одной задаче за раз, проверяет ответ и соответственно корректирует свой подход. Точно так же данный метод обновляет параметры модели машинного обучения для каждой выборки в обучающих данных и подход за подходом минимизирует ошибку.

«Стохастическая» часть названия относится к тому факту, что алгоритм использует случайную выборку из обучающих данных для обновления параметров модели на каждой итерации. Это делает алгоритм быстрее, чем традиционный градиентный спуск, который обновляет параметры, используя весь набор обучающих данных (рисунок 12).

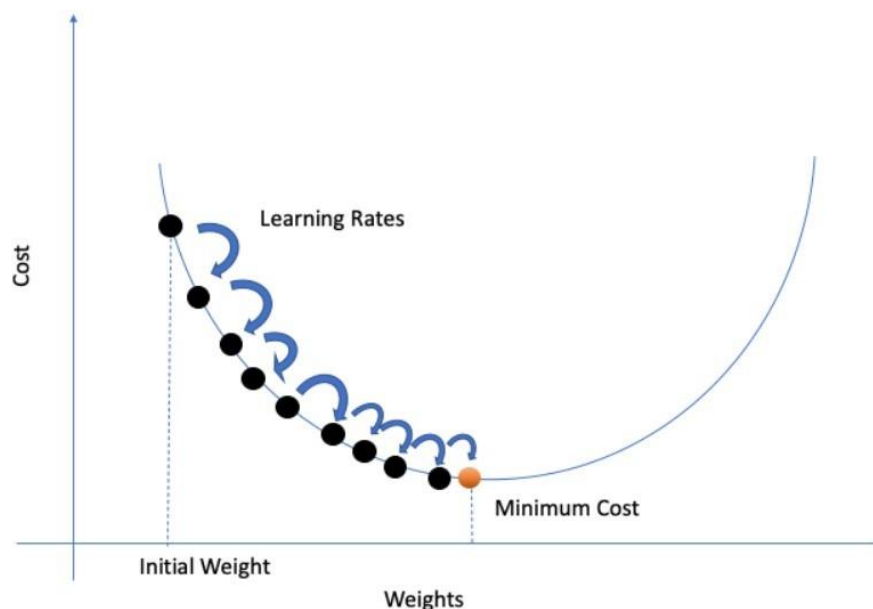


Рисунок 12 – Градиентный спуск

В основе данного исследования лежит классификация полученных данных после проведенного тестирования для определения уровня

обученности студента, поэтому определим такое понятие, как персептрон, который наилучшим образом справляется с задачей классификации. Персептрон представляет собой простейший вид нейронных сетей. После обучения персептрон готов работать в режиме распознавания или обобщения. В основе лежит математическая модель восприятия информации мозгом, состоящая из сенсоров, ассоциативных и реагирующих элементов. Нейронные сети часто обучаются стохастически, то есть на разных итерациях используются разные части данных. Это определяется, как минимум, двумя причинами: во-первых, наборы данных, используемые для обучения, часто очень большие, чтобы хранить их полностью в оперативной памяти и/или производить вычисления эффективно; во-вторых, оптимизируемая функция обычно невыпуклая.

Таким образом, использование разных частей данных на каждой итерации может помочь от застревания модели в локальном минимуме. Кроме того, обучение нейронных сетей обычно производится с помощью градиентных методов первого порядка, так как из-за большого количества параметров в нейронной сети невозможно эффективно применять методы более высоких порядков. Стандартным методом обучения нейронных сетей является метод стохастического градиентного спуска (SGD). Однако он может расходиться или сходиться очень медленно, если шаг обучения настроен недостаточно аккуратно. Поэтому существует много альтернативных методов, с целью ускорения сходимости обучения и избавить пользователя от необходимости тщательной настройки гиперпараметров [36].

Эти методы часто более эффективно вычисляют градиенты и адаптивно изменяют шаг обучения по итерациям. Суть данного алгоритма отражается в следующем уравнении:

$$\omega_1 \leftarrow \omega_1 - \eta \frac{\partial L}{\partial \omega_1},$$

где η является константой, которая называется оценкой обучения, которая контролирует скорость обучения.

Необходимо найти производную функции ошибки по весам и смещениям нейронов для персептрона, если она существует (дифференцируема).

Метод обратного распространения ошибки позволяет выполнить процедуру коррекции весов по направлению от выхода ко входам нейронов путем использования частных производных. Качество обучения можно контролировать путем анализа ошибок сети на ее выходе при сравнении с истинными значениями данных (рисунок 13). В процессе обучения, как правило, эти ошибки уменьшаются по экспоненте, но может наступить момент, когда ошибка сети начинает расти. С этого момента фиксируется так называемый эффект переобучения сети, когда процесс обучения целесообразно остановить. Для анализа качества обучения используется специальный раздел данных, который носит название тестовый набор. Это часть данных, аналогичная тренировочному набору, но не использованная в процессе обучения сети. То есть эта часть данных сети не знакома, и тестирование качества обучения при предъявлении новых данных является более корректным.

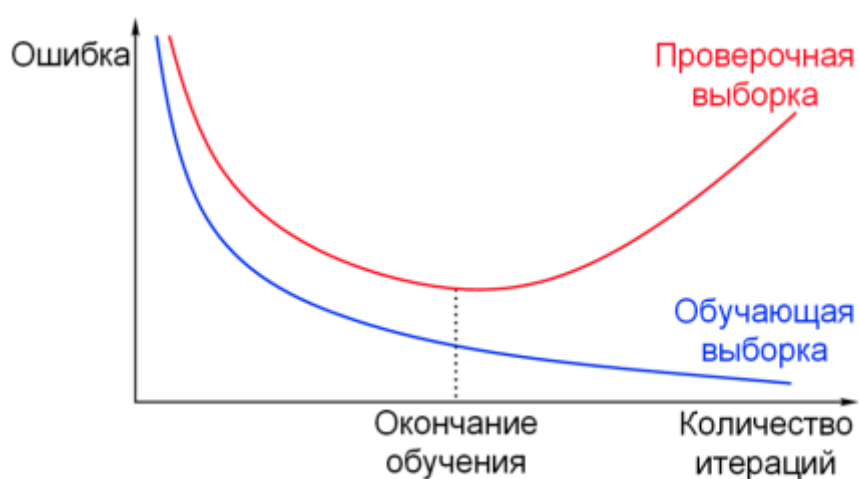


Рисунок 13 – Графическое представление метода обратного распространения ошибки

При вычислении градиента для персептрона по разным параметрам имеют общие черты, вычисляя производные для весов нейронов на втором слое, мы будем делать часть вычислений совпадающих с вычислениями производных для весов нейронов третьего слоя и т.п. Это не случайно и определяется структурой связей между нейронами, и такой факт можно использовать для облегчения расчета производных функции ошибки по параметрам. Метод получил названия метод обратного распространения ошибки (backpropagation) (рисунок 14).

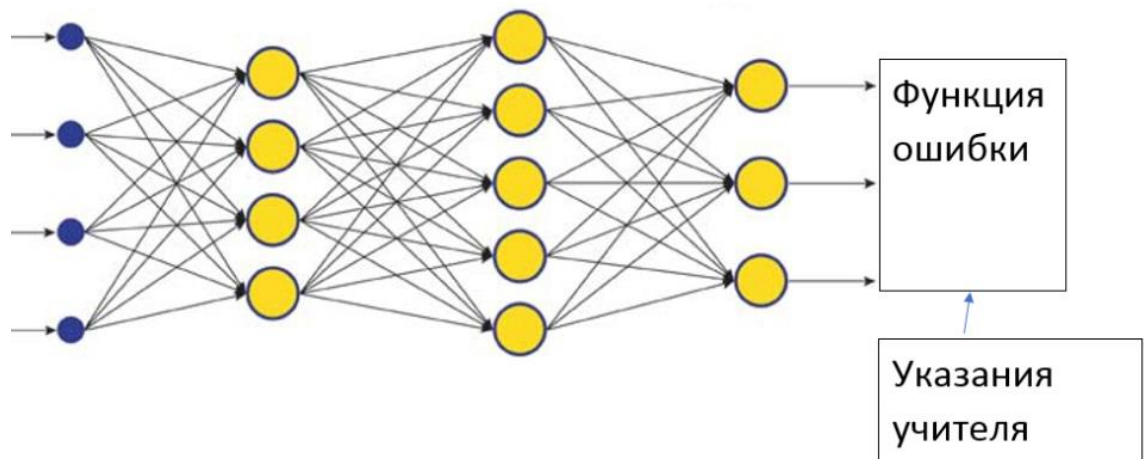


Рисунок 14 - Метод обратного распространения ошибки

Функция ошибки $E(y_1^M, y_2^M, y_3^M, \dots, t)$ есть функция от выходов персептрона, обозначим их как y_i^M , i -ый выход в последнем слое M (это не степень, а индекс), и указаний учителя t . От входов, весов и прочего она зависит только косвенно, через выходы персептрона.

Выход y_i^M любого нейрона m -ого слоя зависит от:

- входов в него, т.е. выходов нейронов предыдущего $m-1$ слоя (для первого слоя - от входов в сеть)
- параметров нейрона (веса\смещения, функция активации).

От входов сети, параметров нейронов на предыдущих слоях такие выходы зависят лишь косвенно, через выходы нейронов предыдущего слоя (кроме первого слоя, который напрямую зависит от входов). От выходов и параметров нейронов на этом же или следующих слоях такие выходы не зависят.

Это справедливо для любого слоя нейронов m :

$$\begin{aligned} & y_1^m(y_1^{m-1}, y_2^{m-1}, y_3^{m-1} \dots, W_1^m); \\ & y_2^m(y_1^{m-1}, y_2^{m-1}, y_3^{m-1} \dots, W_2^m); \\ & y_3^m = \dots \end{aligned}$$

(для первого слоя вместо выходов y_i^0 берем входы сети x_i).

Вид зависимости определен структурой связей персептрона.

Например, выход первого нейрона предпоследнего слоя является аргументом (входом) для выходов всех нейронов последнего слоя, которые являются аргументами функции ошибки.

Тогда по «цепному» правилу дифференцирования для любого слоя нейронов, кроме последнего, можем найти производные функции ошибки по выходам нейронов в нем через производные по всем выходам нейронов в следующем слое. Производная функции ошибки имеет вид:

$$\frac{\partial E}{\partial y_i^m} = \sum_{j=1}^{N_{m+1}} \frac{\partial E}{\partial y_j^{m+1}} \cdot \frac{\partial y_j^{m+1}}{\partial y_i^m}$$

где второй множитель в сумме это производные выхода нейрона по его входу, а первый множитель - производные ошибки по выходам нейронов следующего слоя.

Производные считаются в обратном порядке по выходам нейронов: если знаем для последнего, по формуле пересчитывается для предпоследнего, а зная для предпоследнего, пересчитаем для пред-предпоследнего слоя и т.д.

Вес (смещение) влияет в первую очередь на выход своего нейрона, которому он принадлежит, а потом уж этот выход влияет на выходы нейронов в следующем слое и функцию ошибки. По правилу дифференцирования сложной функции получаем для любого веса нейрона (и смещения):

$$\frac{\partial E}{\partial \omega_{ij}^m} = \frac{\partial E}{\partial y_j^m} \cdot \frac{\partial y_j^m}{\partial \omega_{ij}^m}$$

где первый множитель берем из предыдущей формулы, второй - производная выхода нейрона по одному из его весов - считается легко (а ну-ка посчитайте).

Эти две формулы составляют суть метода обратного распространения ошибки, в котором производная ошибки пересчитывается итеративно по слоям, начиная с последнего.

Таким образом работает обучение нейронных сетей. Метод обратного распространения ошибки для расчета производных очень универсален, для других сетей, в которых нейроны связаны по-другому, он также может быть применен, только надо учитывать, как нейроны влияют друг на друга. И даже не для нейронных сетей, а для произвольных вычислений, можно реализовать такой метод, так работает автодифференцирование.

Процесс тренировки нейронной сети на основе стохастического градиентного спуска будет выглядеть следующим образом:

- выбирается один пункт из набора данных,
- подсчитываются все частные производные потери по весу или смещению,
- применяется уравнение метод обратного распространения ошибки, для обновления каждого веса и смещения,
- возврат к первому пункту.

У данного алгоритма можно выделить ряд преимуществ:

- ошибка на каждом шаге считается быстро, веса меняются сразу же, что очень сильно ускоряет обучение. Обучение может сойтись ещё до того, как был выполнен единственный проход по всем тренировочным примерам. К тому же не надо хранить всю обучающую выборку в памяти,
- стохастический градиентный спуск работает «более случайно», чем обычный, и поэтому можно надеяться, что он не остановится в маленьких локальных минимумах. Пакетный же спуск хорош для строго выпуклых

функций, потому что уверенно стремится к минимуму глобальному или локальному,

– подходит для онлайн-обучения, т.е. в случаях, когда обучающая выборка постоянно обновляется.

При обучении нейросети типа «персептрон» требуется изменять весовые коэффициенты сети так, чтобы минимизировать среднюю ошибку на выходе нейронной сети при подаче на вход последовательности обучающих входных данных. Формально, чтобы сделать всего один шаг по методу градиентного спуска (сделать всего одно изменение параметров сети), необходимо подать на вход сети последовательно абсолютно весь набор обучающих данных, для каждого объекта обучающих данных вычислить ошибку и рассчитать необходимую коррекцию коэффициентов сети (но не делать эту коррекцию), и уже после подачи всех данных рассчитать сумму в корректировке каждого коэффициента сети (сумма градиентов) и произвести коррекцию коэффициентов «на один шаг». Показано, что при большом наборе обучающих данных алгоритм будет работать крайне медленно, поэтому на практике часто производят корректировку коэффициентов сети после каждого элемента обучения. Здесь значение градиента аппроксимируются градиентом функции стоимости, вычисленном только на одном элементе обучения показанного стохастического градиентного спуска является одной из форм стохастического приближения. Теория стохастических приближений даёт условия сходимости метода стохастического градиентного спуска.

2.4 Описание разработанной программы

Для создания, обучения и дальнейшего использования нейронной сети, используемой для определения уровня знаний студентов, будем использовать PyTorch. Это широко используемая библиотека машинного обучения с открытым исходным кодом, разработанная командой Facebook AI Research. Он основан на Torch, который представляет собой научный вычислительный фреймворк с широкой поддержкой алгоритмов машинного обучения. PyTorch

разработанный таким образом, чтобы быть гибким и эффективным, что делает его идеальным для исследовательских и производственных сред.

PyTorch поддерживает широкий спектр алгоритмов машинного обучения, включая глубокое обучение. Он предоставляет простой и удобный интерфейс для построения и обучения нейронных сетей. PyTorch также поддерживает автоматическое дифференцирование, что упрощает вычисление градиентов и выполнение обратного распространения.

Прежде чем использовать модель PyTorch, были подготовлены данные. Это включало в себя загрузку данных, их предварительную обработку и разделение на обучающий и тестовый наборы.

Далее при подготовке данных была проведена их загрузка в память. PyTorch предоставляет несколько утилит загрузки данных, которые упрощают этот процесс.

После того, как данные были загружены, они были предварительно обработаны. Это включало в себя нормализацию данных и преобразование их в формат, который может использоваться моделью PyTorch.

Перед созданием модели PyTorch была определена сетевая архитектуры. Это включало в себя создание класса, который наследует от `.torch.nn.Module`

Класс должен определять уровни сети с помощью различных модулей PyTorch.

После определения архитектуры сети, было произведено обучение модели. Это включало в себя определение функции потерь и оптимизатора, а затем запуск цикла обучения.

Функция потерь используется для измерения разницы между прогнозируемым и фактическим значениями. PyTorch предоставляет несколько функций потерь, таких как

`.torch.nn.MSELoss` `torch.nn.CrossEntropyLoss`

Оптимизатор используется для обновления весовых коэффициентов модели во время обучения. PyTorch предоставляет несколько оптимизаторов, таких как `.torch.optim.Adam` `torch.optim.SGD`.

Цикл обучения включает в себя итерацию обучающих данных и обновление весовых коэффициентов модели с помощью оптимизатора. После каждой эпохи модель оценивается на тестовых данных для измерения ее производительности.

После того, как модель обучена, мы можем использовать ее для прогнозирования результатов. Для этого необходимо загрузить сохраненную модель и использовать ее для прогнозирования результатов на основе новых данных.

Чтобы загрузить сохраненную модель, мы можем использовать функцию. Эта функция загружает сохраненный словарь состояний модели в память `.torch.load`

```
model = torch.load('model.pt').
```

Чтобы предсказать результаты с помощью загруженной модели, нам нужно пропустить входные данные через модель и получить прогнозируемый выход.

```
y_pred = model(x_test).
```

Переменная должна быть тензором PyTorch, содержащим входные данные. Переменная будет содержать прогнозируемые выходные данные `x_testy_pred`.

Алгоритм обучения нейронной сети реализован так, что функционал ошибки сети вычисляется по всей обучающей выборке (аналогично классическому градиентному спуску).

Был реализован алгоритм стохастического обучения (аналогично стохастическому градиентному спуску).

Предлагаема нейронная сеть, осуществляющую классификацию трех уровней успеваемости студентов по всем 6 имеющимся признакам.

Была подготовлена обучающая выборка следующей структуры, представленной в таблице 3.

Таблица 3 – Обучающая выборка

id	hardtime	harderr	middletime	middleerr	lighttime	lighterr	skill
1001	5	0	3	0	1	0	продвинутый
1002	8	0	4	0	1	0	продвинутый
1003	10	0	8	1	4	1	достаточный
1004	8	0	5	1	1	0	продвинутый
1005	11	1	8	2	4	1	достаточный
1006	10	1	3	0	2	0	продвинутый
1007	10	0	8	1	4	1	достаточный
1008	11	1	8	2	4	1	достаточный
1009	7	0	5	0	1	0	продвинутый
1010	8	0	7	1	2	0	продвинутый
1011	10	1	8	2	4	1	достаточный
1012	11	1	8	2	4	1	достаточный
1013	7	0	4	0	1	0	продвинутый
1014	10	0	8	1	3	1	достаточный

Для каждого объекта (студента) описано 6 параметров, а также идентификационный номер. В зависимости от совокупности факторов, объект будет отнесен к одному из уровней:

- продвинутый,
- достаточный,
- удовлетворительный.

Ниже представлен листинг модуля для обучения Нейронной сети.

```
import pandas as pd
import numpy as np
from neural import MLPptorch
import torch
import torch.nn as nn

# функция обучения
def train(x, y, num_iter):
    for i in range(0, num_iter):
        pred = net.forward(x)
        loss = lossFn(pred, y)
        loss.backward()
        optimizer.step()
        if i%(num_iter/10)==0:
```

```

        print('Ошибка на ' + str(i) + ' итерации: ', loss.item())
    return loss.item()

df = pd.read_csv('концепт.csv', sep=';', encoding='cp1251',
                header=0, index_col=0)
df = df.iloc[np.random.permutation(len(df))]

X = df.iloc[0:90, 0:6].values
y = df.iloc[0:90, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2,
          'удовл':3}).values.reshape(-1,1)
Y = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)

X_test = df.iloc[91:100, 0:6].values
y = df.iloc[91:100, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2,
          'удовл':3}).values.reshape(-1,1)
Y_test = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y_test[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)

inputSize = X.shape[1] # количество входных сигналов равно
                        количеству признаков задачи
hiddenSizes = 20 # задаем число нейронов скрытого слоя
outputSize = Y.shape[1] if len(Y.shape) else 1 # количество
                                                  выходных сигналов равно количеству классов задачи

net = MLPptorch(inputSize,hiddenSizes,outputSize)
lossFn = nn.MSELoss()

optimizer = torch.optim.SGD(net.parameters(), lr=0.05)

# как работает регуляризация

```

```

# optimizer = torch.optim.SGD(net.parameters(), lr=0.05,
    weight_decay=0.1)

loss_ = train(torch.from_numpy(X.astype(np.float32)),
    torch.from_numpy(Y.astype(np.float32)), 5000)

# for name, param in net.named_parameters():
#     print(name, param)
pred =
    net.forward(torch.from_numpy(X.astype(np.float32))).de
    tach().numpy()
err = sum(abs((pred>0.5)-Y))
print(err)
pred
=
net.forward(torch.from_numpy(X_test.astype(np.float32))).detach(
).numpy()
err = sum(abs((pred>0.5)-Y_test))
print(err)

```

В программе используется отдельно описанный модуль neural. Его листинг представлен ниже.

```

import numpy as np
import torch
import torch.nn as nn

class Perceptron:
    def __init__(self, inputSize, hiddenSizes, outputSize):

        self.Win = np.zeros((1+inputSize,hiddenSizes))
        self.Win[0,:] = (np.random.randint(0, 3, size =
            (hiddenSizes)))
        self.Win[1:,:] = (np.random.randint(-1, 2, size =
            (inputSize,hiddenSizes)))

```

```

        self.Wout = np.random.randint(0, 2, size =
            (1+hiddenSizes,outputSize)).astype(np.float64)
        #self.Wout = np.random.randint(0, 3, size =
            (1+hiddenSizes,outputSize))

    def predict(self, Xp):
        hidden_predict = np.where((np.dot(Xp,
self.Win[1:,:]) + self.Win[0,:]) >= 0.0, 1, -1).astype(np.float64)
        out = np.where((np.dot(hidden_predict,
self.Wout[1:,:]) + self.Wout[0,:]) >= 0.0, 1, -
1).astype(np.float64)
        return out, hidden_predict

    def train(self, X, y, n_iter=5, eta = 0.01):
        for i in range(n_iter):
            print(self.Wout.reshape(1, -1))
            for xi, target, j in zip(X, y,
range(X.shape[0])):
                pr, hidden = self.predict(xi)
                self.Wout[1:] += ((eta * (target - pr)) *
hidden).reshape(-1, 1)
                self.Wout[0] += eta * (target - pr)
            return self

class MLP:
    def __init__(self, inputSize, outputSize,
learning_rate=0.1, hiddenSizes = 5):
        # инициализируем нейронную сеть
        # веса инициализируем случайными числами, но теперь
будем хранить их списком
        self.weights = [
            np.random.uniform(-2, 2,
size=(inputSize,hiddenSizes)), # веса скрытого слоя
            np.random.uniform(-2, 2,
size=(hiddenSizes,outputSize)) # веса выходного слоя

```

```

    ]
    self.learning_rate = learning_rate
    self.layers = None

    # сигмоида
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    # нам понадобится производная от сигмоиды при вычислении
градиента
    def derivative_sigmoid(self, x):
        return self.sigmoid(x) * (1 - self.sigmoid(x))

    # прямой проход
    def feed_forward(self, x):
        input_ = x # входные сигналы
        hidden_ = self.sigmoid(np.dot(input_,
self.weights[0]))# выход скрытого слоя = сигмоида(входные
сигналы*веса скрытого слоя)
        output_ = self.sigmoid(np.dot(hidden_,
self.weights[1]))# выход сети (последнего слоя) = сигмоида(выход
скрытого слоя*веса выходного слоя)

        self.layers = [input_, hidden_, output_]
        return self.layers[-1]

    # backprop собственной персоной
    # на вход принимает скорость обучения, реальные ответы,
предсказанные сетью ответы и выходы всех слоев после прямого
прохода
    def backward(self, target):

        # считаем производную ошибки сети
        err = (target - self.layers[-1])

```

```

        # прогоняем производную ошибки обратно ко входу,
        считая градиенты и корректируя веса
        # для этого используем chain rule
        # цикл перебирает слои от последнего к первому
        for i in range(len(self.layers)-1, 0, -1):
            # градиент слоя = ошибка слоя * производную
            функции активации * на входные сигналы слоя

            # ошибка слоя * производную функции активации
            err_delta = err *
            self.derivative_sigmoid(self.layers[i])

            # пробрасываем ошибку на предыдущий слой
            err = np.dot(err_delta, self.weights[i - 1].T)

            # ошибка слоя * производную функции активации *
            на входные сигналы слоя
            dw = np.dot(self.layers[i - 1].T, err_delta)

            # обновляем веса слоя
            self.weights[i - 1] += self.learning_rate * dw

# функция обучения чередует прямой и обратный проход
def train(self, x_values, target):
    self.feed_forward(x_values)
    self.backward(target)

# функция предсказания возвращает только выход последнего
слоя
def predict(self, x_values):
    return self.feed_forward(x_values)

class MLPptorch(nn.Module):
    # как и раньше для инициализации на вход нужно подать
    размеры входного, скрытого и выходного слоев

```

```

def __init__(self, in_size, hidden_size, out_size):
    nn.Module.__init__(self)
    # nn.Sequential - контейнер модулей
    # он последовательно объединяет и позволяет запускать
их одновременно
    self.layers = nn.Sequential(nn.Linear(in_size,
hidden_size), # слой линейных сумматоров
nn.Sigmoid(), # функция активации
nn.Linear(hidden_size, out_size),
nn.Sigmoid(),
)

# прямой проход
def forward(self, x):
    return self.layers(x)

```

В данном модуле описан класс **Perceptron**, который моделирует работы персептрона.

```

class Perceptron:
    def __init__(self, inputSize, hiddenSizes, outputSize):

        self.Win = np.zeros((1+inputSize,hiddenSizes))
        self.Win[0,:] = (np.random.randint(0, 3, size =
(hiddenSizes)))
        self.Win[1:,:] = (np.random.randint(-1, 2, size =
(inputSize,hiddenSizes)))

        self.Wout = np.random.randint(0, 2, size =
(1+hiddenSizes,outputSize)).astype(np.float64)
        #self.Wout = np.random.randint(0, 3, size =
(1+hiddenSizes,outputSize))

    def predict(self, Xp):
        hidden_predict = np.where((np.dot(Xp,
self.Win[1:,:]) + self.Win[0,:]) >= 0.0, 1, -1).astype(np.float64)

```

```

        out = np.where((np.dot(hidden_predict,
self.Wout[1:,:]) + self.Wout[0,:]) >= 0.0, 1, -
1).astype(np.float64)
        return out, hidden_predict
    def train(self, X, y, n_iter=5, eta = 0.01):
        for i in range(n_iter):
            print(self.Wout.reshape(1, -1))
            for xi, target, j in zip(X, y,
range(X.shape[0])):
                pr, hidden = self.predict(xi)
                self.Wout[1:] += ((eta * (target - pr)) *
hidden).reshape(-1, 1)
                self.Wout[0] += eta * (target - pr)
        return self

```

В данном классе реализованы методы:

- конструктор `__init__`, который принимает три параметра, связанные с количеством нейронов во входном, скрытом и выходном слоях,
- метод, реализующий составление прогноза `predict`,
- метод, реализующий обучение `train`,
- реализован класс MLP, который реализует модель многослойного персептрона.

```

class MLP:
    def __init__(self, inputSize, outputSize,
        learning_rate=0.1, hiddenSizes = 5):
        self.weights = [
            np.random.uniform(-2, 2, size = (inputSize,
                hiddenSizes)), # веса скрытого слоя
            np.random.uniform(-2, 2, size = (hiddenSizes,
                outputSize)) # веса выходного слоя
        ]
        self.learning_rate = learning_rate
        self.layers = None
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
    def derivative_sigmoid(self, x):

```

```

        return self.sigmoid(x) * (1 - self.sigmoid(x))
def feed_forward(self, x):
    input_ = x # входные сигналы
    hidden_ = self.sigmoid(np.dot(input_,
        self.weights[0])) # выход скрытого слоя =
        сигмоида(входные сигналы*веса скрытого слоя)
    output_ = self.sigmoid(np.dot(hidden_,
        self.weights[1]))# выход сети (последнего слоя) =
        сигмоида(выход скрытого слоя*веса выходного
        слоя)
    self.layers = [input_, hidden_, output_]
    return self.layers[-1]
def backward(self, target):
    err = (target - self.layers[-1])
    for i in range(len(self.layers)-1, 0, -1):
        err_delta = err *
self.derivative_sigmoid(self.layers[i])
        err = np.dot(err_delta, self.weights[i - 1].T)
        dw = np.dot(self.layers[i - 1].T, err_delta)
        self.weights[i - 1] += self.learning_rate * dw
def train(self, x_values, target):
    self.feed_forward(x_values)
    self.backward(target)
    def predict(self, x_values):
        return self.feed_forward(x_values)

```

В данном классе реализованы одни из основных методов:

- конструктор `__init__`, который инициализирует нейронную сеть. В скрытом слое изначально предполагается 5 нейронов. Начальные значения предлагается инициализировать случайными значениями,
- метод `sigmoid`, который реализует функцию сигмоиды. Также имеются метод, задающий производную от сигмоиды для вычисления градиента,
- метод `feed_forward`, который реализует прямой проход,

- метод `backward`, который реализует обратный проход,
- метод обучения `train`, который сочетает прямой и обратный ходы,
- метод предсказания `predict`, который возвращает только выход последнего слоя,
- также имеется специальный класс `MLPtorch`, который также реализует модель персептрона.

Для организации обучения использовалась среда Google Collab (рисунок 15).

```
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 26 20:24:56 2021

@author: AM4
"""

import pandas as pd
import numpy as np
from neural import MLPtorch
import torch
import torch.nn as nn

# функция обучения
def train(x, y, num_iter):
    for i in range(0, num_iter):
        pred = net.forward(x)
        loss = lossFn(pred, y)
        loss.backward()
        optimizer.step()
        if i%(num_iter/10)==0:
            print("Ошибка на " + str(i) + " итерации: ", loss.item())
    return loss.item()

df = pd.read_csv('концепт.csv', sep=';', encoding='cp1251', header=0, index_col=0)
df = df.iloc[np.random.permutation(len(df))]

X = df.iloc[0:90, 0:6].values
y = df.iloc[0:90, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2, 'удовл': 3}).values.reshape(-1,1)
Y = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)

X_test = df.iloc[91:100, 0:6].values
y = df.iloc[91:100, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2, 'удовл': 3}).values.reshape(-1,1)
Y_test = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y_test[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)
```

Рисунок 15 – Вид среды

Как видно из представленного рисунка 16, обучение сети прошло успешно. Чем выше количество итераций, тем меньше ошибка. Это свидетельствует о том, что выбранный метод градиентного спуска выбран верно, было правильно найдено направление, в котором функция ошибки уменьшается.

```

Ошибка на 0 итерации: 0.23995858430862427
Ошибка на 500 итерации: 0.0011944215511903167
Ошибка на 1000 итерации: 1.0720351383497473e-05
Ошибка на 1500 итерации: 5.947197734279541e-10
Ошибка на 2000 итерации: 1.5773510497751886e-08
Ошибка на 2500 итерации: 5.668808521555602e-13
Ошибка на 3000 итерации: 2.194360238617157e-18
Ошибка на 3500 итерации: 9.125558299674437e-21
Ошибка на 4000 итерации: 3.9747741207970396e-23
Ошибка на 4500 итерации: 1.7290121432951545e-25
[0. 0. 0.]
[0. 0. 0.]

```

Рисунок 16 – Результат обучения

Для дальнейшего использования обученной нейронной сети она была сохранена, для добавления новых данных о студенте, прошедшем тестирование, был создан тензор (рисунок 17).

```

MLPptorch(
  (layers): Sequential(
    (0): Linear(in_features=6, out_features=20, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=20, out_features=3, bias=True)
    (3): Sigmoid()
  )
)

```

```

pred = net.forward(torch.from_numpy(X_test.astype(np.float32))).detach().numpy()
err = sum(abs((pred>0.5)-Y_test))
print(err)
#torch.save(net.state_dict(), 'my_model.pth')
#print(net)
#net = torch.load('my_model.pth')
x_test=torch.tensor([19.0,2.0,14.0, 2.0,5.0, 1.0])
y_pred=net(x_test)
print(y_pred)

```

Рисунок 17 – Вид сохраненной сети

Как видно нейронная сеть (рисунок 18, 19) выдает правильно определенный уровень студента, прошедшего тестирование по новой теме, из чего можно сделать вывод, что поставлены цели достигнуты, задачи успешно решены.

```

pred = net.forward(torch.from_numpy(X_test.astype(np.float32))).detach().numpy()
err = sum(abs((pred>0.5)-Y_test))
print(err)
#torch.save(net.state_dict(), 'my_model.pth')
#print(net)
#net = torch.load('my_model.pth')
x_test=torch.tensor([8.0,0.0,4.0, 0.0,1.0,0.0])
y_pred=net(x_test)
print(y_pred)

```

```

Ошибка на 0 итерации: 0.24745330214500427
Ошибка на 500 итерации: 0.007827983237802982
Ошибка на 1000 итерации: 0.003763186512514949
Ошибка на 1500 итерации: 0.003700738772749901
Ошибка на 2000 итерации: 0.003701336681842804
Ошибка на 2500 итерации: 0.0036996682174503803
Ошибка на 3000 итерации: 0.0036913815420120955
Ошибка на 3500 итерации: 0.0007404039497487247
Ошибка на 4000 итерации: 2.8695795739608343e-10
Ошибка на 4500 итерации: 4.152572383175901e-11
[0. 0. 0.]
[0. 0. 0.]
tensor([1., 0., 0.], grad_fn=<SigmoidBackward0>)

```

Рисунок 18 – Определение уровня обученности студента

```

pred = net.forward(torch.from_numpy(X_test.astype(np.float32))).detach().numpy()
err = sum(abs((pred>0.5)-Y_test))
print(err)
torch.save(net.state_dict(), 'my_model.pth')
print(net)
x_test=torch.tensor([13.0,1.0,9.0, 1.0,6.0, 0.0])
y_pred=net(x_test)
print(y_pred)

```

```

Ошибка на 0 итерации: 0.2618524730205536
Ошибка на 500 итерации: 0.011185229755938053
Ошибка на 1000 итерации: 0.007379322312772274
Ошибка на 1500 итерации: 0.0074109421111643314
Ошибка на 2000 итерации: 0.007407189346849918
Ошибка на 2500 итерации: 0.0037042396143078804
Ошибка на 3000 итерации: 0.003703707130625844
Ошибка на 3500 итерации: 0.0037037080619484186
Ошибка на 4000 итерации: 0.0037196900229901075
Ошибка на 4500 итерации: 0.003706328570842743
[0. 1. 0.]
[0. 0. 0.]
MLPtorch(
  (layers): Sequential(
    (0): Linear(in_features=6, out_features=20, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=20, out_features=3, bias=True)
    (3): Sigmoid()
  )
)
tensor([0., 1., 0.], grad_fn=<SigmoidBackward0>)

```

Рисунок 19 – Определение уровня обученности студента

ЗАКЛЮЧЕНИЕ

В диссертации решена задача, заключающаяся в разработке системы, позволяющей использовать нейронные сети и их возможности при обучении математике с целью повышения уровня знаний студентов.

Основные результаты работы состоят в следующем:

- полученные данные, соответствующие трем уровням обученности (удовлетворительный, достаточный, продвинутый) с помощью компьютерного тестирования и разноуровневых задач,
- проведен анализ имеющихся архитектур нейронных сетей для классификации уровня обученности студента. В результате была выбрана архитектура многослойного персептрона,
- был реализован алгоритм стохастического обучения нейронной сети, которая осуществляет классификацию трех уровней успеваемости студентов по 6 имеющимся признакам,
- разработана и реализована нейронная сеть, определяющая уровень знаний студентов с целью повышения их познавательной и мыслительной активности. Успешно обученная нейронная сеть протестирована и показала свою работоспособность.

Практической значимостью данного исследования является возможность использования разработанной нейронной сети в образовательном процессе с целью повышения эффективности процесса обучения студентов при изучении различных дисциплин.

Таким образом, цель работы достигнута, все поставленные задачи решены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Российская педагогическая энциклопедия: В 2 т./Гл. ред. В.В. Давыдов – М.: Большая Рос. энциклопедия, 2002. – Т. 1.- А-М.
2. Унт И. Э. Индивидуализация и дифференциация обучения/И.Э. Унт. – М.: Педагогика, 2003. – 192 с.
3. Монахов В.М. Дифференциация обучения в средней школе/В.М. Монахов, В.А. Орлов, В.В. Фирсов // Советская педагогика. – 1990. – №8. – С.42-47
- 4.Архангельский С.И. Некоторые новые задачи высшей школы и требования к педагогическому мастерству/С.И. Архангельский. – М., 2012. – 180 с.
5. Пичугина П.Г. Профессиональная направленность обучения математике студентов медицинских вузов. Дис. канд. пед. наук: 13.00.02 /П.Г. Пичугина. Пенза, 2004. – 142
6. Дорофеев Г.В. Дифференциация в обучении математике/Г.В. Дорофеев, Л.В. Кузнецова, С.Б. Суворова, В.В. Фирсов// Математика в школе – 1990. – №4. – С. 15 – 20.
7. Гусев В.А. Индивидуализация учебной деятельности учащихся как основа дифференцированного обучения математике в средней школе./ В.А. Гусев// Математика в школе. – 1990. № 4. С.27-31.
8. Захарова И.Г. Информационные технологии в образовании: Учеб. пособие для студ. высш. учеб. заведений/ И.Г. Захарова. – М.: Издательский центр «Академия», 2005. – 192 с.
9. Михеев С.А. Проблема интеграции компьютерного тестирования и традиционных форм педагогического контроля//Высшее образование сегодня. – 2014. - №5. – С. 66-69.
10. Овчаренков Э.А. Совершенствование методов проверки и контроля знаний студентов ВУЗа – один из путей повышения качества учебного процесса//Современные проблемы науки и образования. – 2014. - №4.

11. Родионова Е.В. Тестирование как один из методов оценки персонала: материалы II международной научно-практической конференции «Актуальные проблемы науки, экономики и образования XXI века» (часть 2). – Самара, 2012. – 392 с.

12. Иванушкина Н.В. Критериально-ориентированное тестирование знаний студентов по курсу «Психология и педагогика»: учебное пособие / Н.В. Иванушкина. – Самара: Издательство Самарского университета, 2020. – 184 с.

13. Калугян К.Х. Компьютерная система тестирования знаний как компонент информационной научно-образовательной среды вуза / К.Х. Калугян, С.М. Щербаков // Вестник Академии. – 2005. – № 1 (20). – С. 61-66.

14. Боровкова Т. И. Мониторинг развития системы образования. Часть 1. Теоретические аспекты: учеб. пособие / Т.И. Боровкова, И.А. Морев. – Владивосток: Изд-во Дальневосточного ун-та, 2004. – 150 с.

15. Мирдак Н.П. Роль и место компьютерного тестирования в системе контроля знаний / Н.П. Мирдак, И.В. Шумилина // Педагогические и информационные технологии в образовании: эл. журн., 2009. – вып. 8.

16. Мамай С. П. Методика составления тестовых заданий: Учеб. пособие. Екатеринбург: Изд-во Урал.гос.проф.-пед.ун-та, 2001. 58 с.

17. Пастущак Т.Н., С.С. Соколов, А.А. Рябова Создание электронного курса. Лекция в СДО Moodle. Учебно-методическое пособие – СПб.: СПГУВК, 2012 – 45 с.

18. Ross K. Doing and proving: the place of algorithms and proof in school mathematics / K. Ross // American Mathematics Monthly. – 2008. – Vol. 3. – P. 252– 255.

19. Jiang J. A review and prospect of readable machine proofs for geometry theorems / J. Jiang, J. Zhong // Journal of Systems Science and Complexity. – 2012. – Vol. 25. No 4. – P. 802–820. – DOI 10.1007/s11424-012-2048-3. – EDN YLKCFS

20. Бессмертный И.А. Семантическая паутина и искусственный интеллект / И.А. Бессмертный // Научно-технический вестник

информационных технологий, математики и оптики. – 2009. – №6 (64). – С. 77–83.

21. Барский А.Б. Логические нейронные сети: учебное пособие / А.Б. Барский. – М.: Бином, 2013. – 352 с.

22. Егорова Ю.Н. Факторы и риски успешной самореализации студента в образовательном пространстве вуза / Ю.Н. Егорова, Ю.А. Генварева // Издательский дом «Среда»

23. Егорова Ю.Н. Электронная образовательная среда как средство организации самостоятельной работы студента железнодорожного вуза / Ю.Н. Егорова, Ю.А. Генварева // Вектор науки Тольяттинского государственного университета. Серия: Педагогика, психология. – 2018. – №2 (33). – С. 21–26. – DOI 10.18323/2221-5662-2018-2-21-26. – EDN XSFVQD.

24. Генварева Ю.А. Современные подходы к преподаванию математики в техническом вузе / Ю.А. Генварева, Н.Г. Марченкова // ЦИТИСЭ. – 2023. – №2 (36). – С. 50–57. – DOI 10.15350/2409-7616.2023.2.04. – EDN HORKNW.

25. Генварева Ю.А. Решение профессионально-ориентированных задач по физике и математике как средство формирования профессиональной компетентности будущего инженера / Ю.А. Генварева, Н.Г. Марченкова // ЦИТИСЭ. – 2022. – №4 (34). – С. 171–179. – DOI 10.15350/2409-7616.2022.4.27. – EDN OPGNHI.

26. Попов А.Н. Формирование корпоративных компетенций будущего инженера ресурсами образовательного процесса технического вуза / А.Н. Попов, А.А. Хандримайлов, О.Ю. Малахова // Проблемы современного педагогического образования. – 2022. – №74–2. – С. 188–191. – EDN UFYZUB.

27. Малахова О.Ю. Детерминированность социокультурного самоопределения и саморегуляции в контексте становления личности студента вуза / О.Ю. Малахова, Д.В. Никифоров // Проблемы современного педагогического образования. – 2022. – №75–4. – С. 180–184. – EDN BUSMWK.

28. Курош А.Г. Курс высшей алгебры/А.Г. Курош. – М., 2013.

ПРИЛОЖЕНИЕ А

Тема: «Использование нейронных сетей при обучении высшей
математике в вузе»

Листинг программ

Листинг А.1 - Модуль для обучения Нейронной сети

```
import pandas as pd
import numpy as np
from neural import MLPptorch
import torch
import torch.nn as nn

# функция обучения
def train(x, y, num_iter):
    for i in range(0,num_iter):
        pred = net.forward(x)
        loss = lossFn(pred, y)
        loss.backward()
        optimizer.step()
        if i%(num_iter/10)==0:
            print('Ошибка на ' + str(i) + ' итерации: ',
loss.item())
    return loss.item()

df = pd.read_csv('концепт.csv', sep=';', encoding='cp1251',
header=0, index_col=0)
df = df.iloc[np.random.permutation(len(df))]

X = df.iloc[0:90, 0:6].values
y = df.iloc[0:90, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2,
'удовл':3}).values.reshape(-1,1)
Y = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)

X_test = df.iloc[91:100, 0:6].values
y = df.iloc[91:100, 6]
y = y.map({'продвинутый': 1, 'достаточный': 2,
'удовл':3}).values.reshape(-1,1)
```

```

Y_test = np.zeros((y.shape[0], np.unique(y).shape[0]))
for i in np.unique(y):
    Y_test[:,i-1] = np.where(y == i, 1, 0).reshape(1,-1)

inputSize = X.shape[1] # количество входных сигналов равно
количеству признаков задачи
hiddenSizes = 20 # задаем число нейронов скрытого слоя
outputSize = Y.shape[1] if len(Y.shape) else 1 # количество
выходных сигналов равно количеству классов задачи

net = MLPptorch(inputSize,hiddenSizes,outputSize)
lossFn = nn.MSELoss()

optimizer = torch.optim.SGD(net.parameters(), lr=0.05)

# как работает регуляризация
# optimizer = torch.optim.SGD(net.parameters(), lr=0.05,
weight_decay=0.1)

loss_ = train(torch.from_numpy(X.astype(np.float32)),
               torch.from_numpy(Y.astype(np.float32)), 5000)

# for name, param in net.named_parameters():
#     print(name, param)

pred =
net.forward(torch.from_numpy(X.astype(np.float32))).detach().numpy()

err = sum(abs((pred>0.5)-Y))
print(err)

pred =
net.forward(torch.from_numpy(X_test.astype(np.float32))).detach(
).numpy()

err = sum(abs((pred>0.5)-Y_test))
print(err)

```

Листинг А.2 - Модуль neural

```
import numpy as np
import torch
import torch.nn as nn

class Perceptron:
    def __init__(self, inputSize, hiddenSizes, outputSize):

        self.Win = np.zeros((1+inputSize,hiddenSizes))
        self.Win[0,:] = (np.random.randint(0, 3, size =
(hiddenSizes)))
        self.Win[1:,:] = (np.random.randint(-1, 2, size =
(inputSize,hiddenSizes)))

        self.Wout = np.random.randint(0, 2, size =
(1+hiddenSizes,outputSize)).astype(np.float64)
        #self.Wout = np.random.randint(0, 3, size =
(1+hiddenSizes,outputSize))

    def predict(self, Xp):
        hidden_predict = np.where((np.dot(Xp,
self.Win[1:,:]) + self.Win[0,:]) >= 0.0, 1, -1).astype(np.float64)
        out = np.where((np.dot(hidden_predict,
self.Wout[1:,:]) + self.Wout[0,:]) >= 0.0, 1, -
1).astype(np.float64)
        return out, hidden_predict

    def train(self, X, y, n_iter=5, eta = 0.01):
        for i in range(n_iter):
            print(self.Wout.reshape(1, -1))
            for xi, target, j in zip(X, y,
range(X.shape[0])):
                pr, hidden = self.predict(xi)
                self.Wout[1:] += ((eta * (target - pr)) *
hidden).reshape(-1, 1)
```

```

        self.Wout[0] += eta * (target - pr)
    return self

class MLP:
    def __init__(self, inputSize, outputSize,
learning_rate=0.1, hiddenSizes = 5):
        # инициализируем нейронную сеть
        # веса инициализируем случайными числами, но теперь
будем хранить их списком
        self.weights = [
            np.random.uniform(-2, 2,
size=(inputSize,hiddenSizes)), # веса скрытого слоя
            np.random.uniform(-2, 2,
size=(hiddenSizes,outputSize)) # веса выходного слоя
        ]
        self.learning_rate = learning_rate
        self.layers = None

    # сигмоида
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))

    # нам понадобится производная от сигмоиды при вычислении
градиента
    def derivative_sigmoid(self, x):
        return self.sigmoid(x) * (1 - self.sigmoid(x))

    # прямой проход
    def feed_forward(self, x):
        input_ = x # входные сигналы
        hidden_ = self.sigmoid(np.dot(input_,
self.weights[0]))# выход скрытого слоя = сигмоида(входные
сигналы*веса скрытого слоя)

```

```

        output_ = self.sigmoid(np.dot(hidden_,
self.weights[1]))# выход сети (последнего слоя) = сигмоида(выход
скрытого слоя*веса выходного слоя)

        self.layers = [input_, hidden_, output_]
        return self.layers[-1]

# backprop собственной персоной
# на вход принимает скорость обучения, реальные ответы,
предсказанные сетью ответы и выходы всех слоев после прямого
прохода
def backward(self, target):

    # считаем производную ошибки сети
    err = (target - self.layers[-1])
    # прогоняем производную ошибки обратно ко входу,
считая градиенты и корректируя веса
    # для этого используем chain rule
    # цикл перебирает слои от последнего к первому
    for i in range(len(self.layers)-1, 0, -1):
        # градиент слоя = ошибка слоя * производную
функции активации * на входные сигналы слоя

        # ошибка слоя * производную функции активации
        err_delta = err *
self.derivative_sigmoid(self.layers[i])

        # пробрасываем ошибку на предыдущий слой
        err = np.dot(err_delta, self.weights[i - 1].T)

        # ошибка слоя * производную функции активации *
на входные сигналы слоя
        dw = np.dot(self.layers[i - 1].T, err_delta)

        # обновляем веса слоя

```

```

        self.weights[i - 1] += self.learning_rate * dw

# функция обучения чередует прямой и обратный проход
def train(self, x_values, target):
    self.feed_forward(x_values)
    self.backward(target)

# функция предсказания возвращает только выход последнего
слоя
def predict(self, x_values):
    return self.feed_forward(x_values)

class MLPptorch(nn.Module):
    # как и раньше для инициализации на вход нужно подать
    размеры входного, скрытого и выходного слоев
    def __init__(self, in_size, hidden_size, out_size):
        nn.Module.__init__(self)
        # nn.Sequential - контейнер модулей
        # он последовательно объединяет и позволяет запускать
их одновременно
        self.layers = nn.Sequential(nn.Linear(in_size,
hidden_size), # слой линейных сумматоров
nn.Sigmoid(), # функция активации
nn.Linear(hidden_size, out_size),
nn.Sigmoid(),
)

# прямой проход
def forward(self, x):
    return self.layers(x)

```

Листинг А.3 - Модуль класса Perceptron

```
class Perceptron:
    def __init__(self, inputSize, hiddenSizes, outputSize):

        self.Win = np.zeros((1+inputSize,hiddenSizes))
        self.Win[0,:] = (np.random.randint(0, 3, size =
(hiddenSizes)))
        self.Win[1:,:] = (np.random.randint(-1, 2, size =
(inputSize,hiddenSizes)))

        self.Wout = np.random.randint(0, 2, size =
(1+hiddenSizes,outputSize)).astype(np.float64)
        #self.Wout = np.random.randint(0, 3, size =
(1+hiddenSizes,outputSize))

    def predict(self, Xp):
        hidden_predict = np.where((np.dot(Xp,
self.Win[1:,:]) + self.Win[0,:]) >= 0.0, 1, -1).astype(np.float64)
        out = np.where((np.dot(hidden_predict,
self.Wout[1:,:]) + self.Wout[0,:]) >= 0.0, 1, -
1).astype(np.float64)
        return out, hidden_predict
    def train(self, X, y, n_iter=5, eta = 0.01):
        for i in range(n_iter):
            print(self.Wout.reshape(1, -1))
            for xi, target, j in zip(X, y,
range(X.shape[0])):
                pr, hidden = self.predict(xi)
                self.Wout[1:] += ((eta * (target - pr)) *
hidden).reshape(-1, 1)
                self.Wout[0] += eta * (target - pr)
            return self

class MLP:
    def __init__(self, inputSize, outputSize,
learning_rate=0.1, hiddenSizes = 5):
```

```

        self.weights = [
            np.random.uniform(-2, 2,
size=(inputSize,hiddenSizes)), # веса скрытого слоя
            np.random.uniform(-2, 2,
size=(hiddenSizes,outputSize)) # веса выходного слоя
        ]
        self.learning_rate = learning_rate
        self.layers = None
    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
    def derivative_sigmoid(self, x):
        return self.sigmoid(x) * (1 - self.sigmoid(x))
    def feed_forward(self, x):
        input_ = x # входные сигналы
        hidden_ = self.sigmoid(np.dot(input_,
self.weights[0])) # выход скрытого слоя = сигмоида(входные
сигналы*веса скрытого слоя)
        output_ = self.sigmoid(np.dot(hidden_,
self.weights[1]))# выход сети (последнего слоя) = сигмоида(выход
скрытого слоя*веса выходного слоя)
        self.layers = [input_, hidden_, output_]
        return self.layers[-1]
    def backward(self, target):
        err = (target - self.layers[-1])
        for i in range(len(self.layers)-1, 0, -1):
            err_delta = err *
self.derivative_sigmoid(self.layers[i])
            err = np.dot(err_delta, self.weights[i - 1].T)
            dw = np.dot(self.layers[i - 1].T, err_delta)
            self.weights[i - 1] += self.learning_rate * dw
    def train(self, x_values, target):
        self.feed_forward(x_values)
        self.backward(target)
    def predict(self, x_values):
        return self.feed_forward(x_values)

```

ПРИЛОЖЕНИЕ Б

Тема: «Использование нейронных сетей при обучении высшей математике в
вузе»

Презентация

«ИСПОЛЬЗОВАНИЕ НЕЙРОННЫХ СЕТЕЙ ПРИ ОБУЧЕНИИ ВЫСШЕЙ МАТЕМАТИКЕ В ВУЗЕ»

Магистрант кафедры вычислительной техники:

Леонова Татьяна Юрьевна

Научный руководитель:

кандидат технических наук, доцент

Артюхин Василий Валерьевич

Пенза, 2024

2

Актуальность,
объект и предмет
исследования

- **Актуальность работы** состоит в потребности совершенствования учебного процесса за счет использования современных цифровых технологий
- **Объект исследования** – процесс обучения высшей математике студентов технических вузов
- **Предмет исследования** – возможности применения нейронной сети для усовершенствования педагогического процесса

3

Цель исследования:

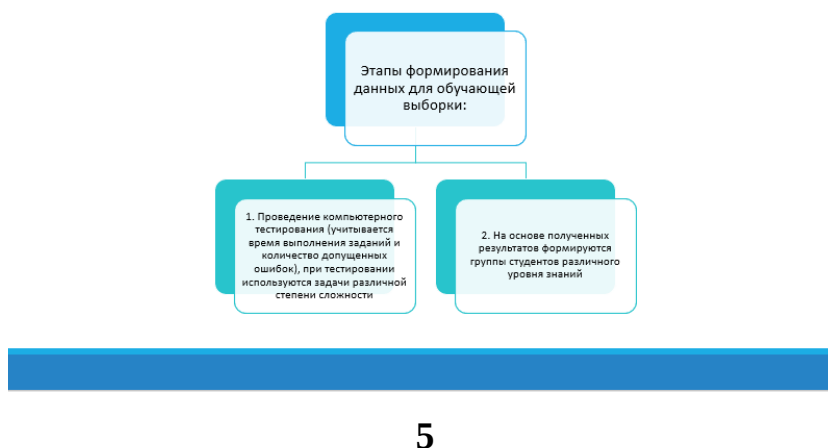
повышение эффективности процесса обучения математике в вузе с помощью использования нейронной сети, которая сможет определить уровень знаний студентов по результатам тестирования при изучении высшей математики

Основные задачи исследования:

- на основе анализа основных понятий нейронных сетей и их классификации, выделить возможности их использования в учебном процессе;
- обоснование выбора внедрения и использования возможностей нейронных сетей с целью повышения качества образовательного процесса;
- разработка системы, позволяющей использовать нейронные сети и их возможности при обучении математике.

4

Первый этап создания нейронной сети



Уровни знаний студентов



Реализация уровневой дифференциации

Основная цель дифференцированного подхода – обучение каждого на уровне его возможностей и способностей, что даёт каждому студенту возможность получить максимальные по его способностям знания и реализовать его личный потенциал

Процесс реализации уровневой дифференциации

составление или подбор дифференцированных заданий, включающие различные приёмы, которые помогают обучающимся самостоятельно справиться с заданием, или связанных с увеличением объёма и сложности задания.

постоянный контроль за результатами работы учащихся, в соответствии с которыми изменяется характер дифференцированных заданий.

Положительные аспекты компьютерного тестирования

1. Повышение эффективности тестирования за счет уменьшения его продолжительности
2. Экономия на раздаточном материале
3. Обработка результатов происходит автоматически
4. Подробная статистика полученных результатов и их объективность

8

Использование нейронных сетей в образовании

Проведение математических вычислений (упрощение, всплывающие подсказки)

Составление «портрета» обучаемого, учитывая различные личностные характеристики

Прогнозирование успеваемости студентов

Создание презентаций, автоматическая проверка выполненного домашнего задания

Создание учебных материалов, в том числе интерактивных

Создание индивидуальных образовательных программ

9

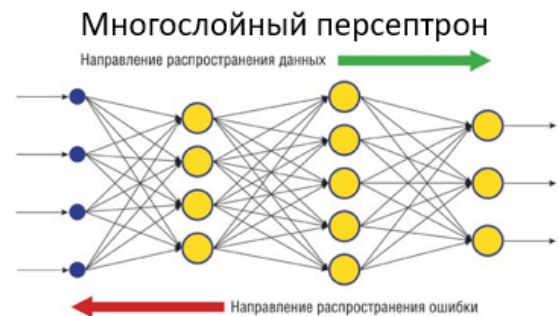
Второй этап создания нейронной сети

Виды нейронных сетей:

Сверточная нейронная сеть
Рекуррентная нейронная сеть
Полносвязная нейронная сеть

Функции активации:

Линейная функция
Функция ReLU
Сигмоида
Гиперболический тангенс



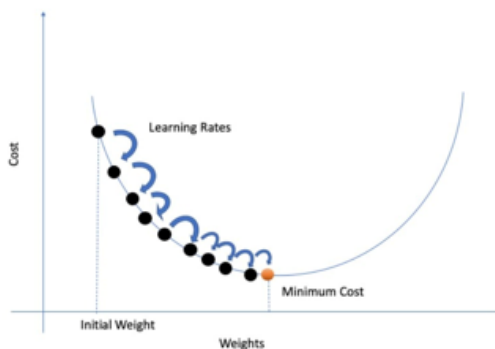
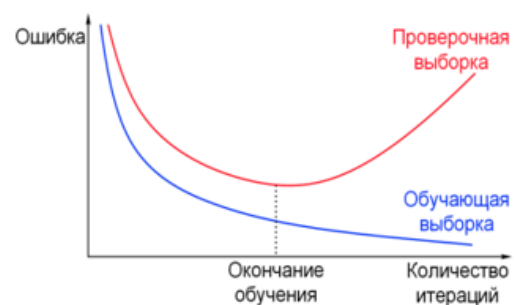
Сигмоидная логистическая функция

$$f(x) = \frac{1}{1 + e^{-x}}$$

10

Третий этап создания нейронной сети

В процессе обучения и контроля за качеством обучения были использованы методы градиентного спуска и обратного распространения ошибки



11



12

Обучение нейронной сети

```
Ошибка на 0 итерации: 0.23995858430862427
Ошибка на 500 итерации: 0.0011944215511903167
Ошибка на 1000 итерации: 1.0720351383497473e-05
Ошибка на 1500 итерации: 5.947197734279541e-10
Ошибка на 2000 итерации: 1.5773510497751886e-08
Ошибка на 2500 итерации: 5.668808521555602e-13
Ошибка на 3000 итерации: 2.194360238617157e-18
Ошибка на 3500 итерации: 9.125558299674437e-21
Ошибка на 4000 итерации: 3.9747741207970396e-23
Ошибка на 4500 итерации: 1.7290121432951545e-25
[0. 0. 0.]
[0. 0. 0.]
... - - -
```

13

Результат работы нейронной сети

```
pred = net.forward(torch.from_numpy(X_test.astype(np.float32))).detach().numpy()
err = sum(abs((pred>0.5)-Y_test))
print(err)
#torch.save(net.state_dict(), 'my_model.pth')
#print(net)
#net = torch.load('my_model.pth')
x_test=torch.tensor([8.0,0.0,4.0, 0.0,1.0,0.0])
y_pred=net(x_test)
print(y_pred)

Ошибка на 0 итерации: 0.24745330214500427
Ошибка на 500 итерации: 0.007827983237802982
Ошибка на 1000 итерации: 0.003763186512514949
Ошибка на 1500 итерации: 0.003700738772749901
Ошибка на 2000 итерации: 0.003701336681842804
Ошибка на 2500 итерации: 0.0036996682174503803
Ошибка на 3000 итерации: 0.0036913815420120955
Ошибка на 3500 итерации: 0.0007404039497487247
Ошибка на 4000 итерации: 2.8695795739608343e-10
Ошибка на 4500 итерации: 4.152572383175901e-11
[0. 0. 0.]
[0. 0. 0.]
tensor([1., 0., 0.], grad_fn=<SigmoidBackward0>)
```

14

Заключение

В результате выполнения работы:

- получены данные, соответствующие трем уровням обученности (удовлетворительный, достаточный, продвинутый) с помощью компьютерного тестирования и разноуровневых задач,
- проведен анализ имеющихся архитектур нейронных сетей для классификации уровня обученности студента. В результате была выбрана архитектура многослойного персептрона,
- был реализован алгоритм стохастического обучения нейронной сети, которая осуществляет классификацию трех уровней успеваемости студентов по 6 имеющимся признакам,
- разработана и реализована нейронная сеть, определяющая уровень знаний студентов с целью повышения их познавательной и мыслительной активности. Успешно обученная нейронная сеть протестирована и показала свою работоспособность.

Таким образом, можно сделать вывод, что поставлена цель работы достигнута, поставленные задачи решены.

Практической значимостью данного исследования является возможность использования разработанной нейронной сети в образовательном процессе с целью повышения эффективности процесса обучения студентов при изучении различных дисциплин.