МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра прикладной математики**

**КУРСОВАЯ РАБОТА**

**РАЗРАБОТКА НЕЙРОСЕТЕВОГО ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ РЕШЕНИЯ ОБРАТНЫХ КОЭФФИЦИЕНТНЫХ ЗАДАЧ**

Работу выполнил\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_О.Ю. Виноградова

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем.

Направленность (профиль) Методы программирования

Научный руководитель

канд. физ-мат. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С.И. Фоменко

Нормоконтролер

преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.С. Троценко

Краснодар

2023

**РЕФЕРАТ**

Курсовая работа 30с., 12 рис., 15 источников.

НЕЙРОСЕТИ, ОБРАТНЫЕ ЗАДАЧИ, ПЕРСЕПТРОН, ОБРАТНОЕ РАСПРОСТРАНЕНИЕ ОШИБКИ

Цель работы: изучить целесообразность применения нейросетевых алгоритмов для решения обратных коэффициентных задач теории упругости, выбрать подходящую архитектуру и алгоритмы обучения искусственной нейронной сети и создать её и оценить работоспособность.

В процессе работы были изучены: основы раздела механики сплошных сред – раздел теории упругости – прямые и обратные задачи, способы их разрешения с помощью современных технологий. Так же нейросетевые алгоритмы, методы их обучений, их архитектуры.

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc135817335)

[1 Теоретические основы работы. 5](#_Toc135817336)

[1.1 Прямые и обратные задачи теории упругости 5](#_Toc135817337)

[1.2 Описание теоретической установки и процесса измерений 7](#_Toc135817338)

[1.3 Архитектура искусственных нейронных сетей 8](#_Toc135817339)

[1.3.1 Обучение нейронной сети 10](#_Toc135817340)

[1.3.2 Алгоритм для решения обратных коэффициентных задач с помощью нейросетей 11](#_Toc135817341)

[2.1 Теоретические основы нейронных сетей 12](#_Toc135817342)

[2.1 Формирование обучающей и тестовых выборок данных 12](#_Toc135817343)

[2.2 Архитектура нейронных сетей 15](#_Toc135817344)

[2.3 Обратное распространение ошибки 19](#_Toc135817345)

[2.3.1 Линейная регрессия 22](#_Toc135817346)

[2.4 Классификация задач для нейронной сети 23](#_Toc135817347)

[2.5 Сравнение результатов при изменении параметров 24](#_Toc135817348)

[Заключние 26](#_Toc135817349)

[Список используемых источников 28](#_Toc135817350)

[Приложение А Код программы 31](#_Toc135817351)

# ВВЕДЕНИЕ

В современном мире пористые материалы играют важную роль в различных областях науки и техники. Они используются в различных отраслях промышленности, таких как нефтяная и газовая промышленность, строительство, энергетика и многих других. Одной из важнейших задач, связанных с пористыми материалами, является измерение их параметров, таких как пористость. Для решения этой задачи часто используются различные методы, включая методы, основанные на использовании волн.

Нейросетевой алгоритм является одним из перспективных методов для решения задач измерения параметров пористых материалов. В данной работе рассматривается использование нейросети для решения обратных коэффициентных задач теории упругости на примере измерения пористости двух материалов, совмещенных на установке. В основе данного метода лежит измерение скорости волн, распространяющихся в пористых материалах

В данной работе рассматривается конкретная установка, состоящая из двух пористых материалов. Один из материалов расположен сверху, имеет фиксированную высоту, второй материал является полупространством. Через верхний материал направляется упругая волна, которая затем принимается приемником. С приемника записываются данные о медленностях (величины, обратные скоростям) и частотах волны. Эти данные отображаются на графике в виде точек. Основной задачей является разработка нейронной сети, которая на основе полученных точек с графика сможет предсказать значения пористости первого и второго материалов.

Целью данного исследования является разработка и обучение нейронной сети для решения обратных коэффициентных задач теории упругости в контексте данной установки с пористыми материалами. Для достижения этой цели ставятся следующие задачи:

Изучение теоретических основ нейронных сетей, включая их архитектуры, методы обучения и применение в задачах анализа данных.

Описание экспериментальной установки, процесса измерений и формирования обучающей и тестовой выборок данных.

Разработка архитектуры нейронной сети, которая будет способна обработать данные с графика и предсказать значения пористости материалов.

Выбор и применение соответствующей функции потерь и оптимизатора для обучения нейронной сети.

Обучение и оценка разработанной модели на основе экспериментально полученных данных.

# Теоретические основы работы.

## Прямые и обратные задачи теории упругости

Для этого класса задач известны причины, требуется найти следствие. В качестве причин могут фигурировать сама математическая модель, начальные условия, коэффициенты дифференциальных операторов, граничные условия, геометрия области.

В различных областях науки, социальных моделях и прочих процессах, принцип функционирования объектов задачи обобщенно выглядит следующим образом:

X – входные данные

Y – система обработки запроса, характеристики и структуры объектов

Z – выходные данные

Тогда, под прямой задачей понимаем необходимость определения выхода Z, если известны вход X и оператор Y.

На основе этих соображений, в качестве обратной задачи понимаем задачу идентификации начального объекта изучения. Тогда неизвестными могут быть оператор Y или часть входной информации X или Y и X.

Проблема определения оператора Y, носит название структурной идентификации системы. Таким образом, прямая задача имеет только один вид, а обратная задача имеет два вида или более, в зависимости от количества информации, известной о 3-х параметрах системы. Для этого класса задач известны следствия, требуется найти причины и определить их по некоторой дополнительной информации об объектах исследования.

Для обратных задач известны следствия, требуется найти причины и определить их по некоторой дополнительной информации об объектах исследования. Эти задачи стали предметом исследований в математике относительно недавно, первые работы в этом направлении относятся к началу XX века, а более интенсивно разработки в этой области математического моделирования начали проводиться в 70 – 80-х годах прошлого века.

Обратная задача используется в самых разных областях механики, физики, геологии и многих других. В качестве примеров обратных задач назовем обнаружение ресурсов, определение материальных констант, граничных условий, диагностика трещин, определение остаточных напряжений и другие. Методы решения подобных задач в настоящее время хорошо развиты. Методы определения неизвестных параметров основаны на анализе реакции системы на определенные воздействия.

К настоящему времени сложилась следующая условная классификация обратных задач:

– ретроспективные обратные задачи (задачи с обращенным временем) – задачи об определении начального состояния объекта исследования (начальных условий) по некоторым функционалам или операторам от решения.

– коэффициентные обратные задачи – задачи определения коэффициентов дифференциальных операторов.

– граничные обратные задачи (задачи об определении граничных условий).

– обратные задачи смешанных типов (неизвестными являются несколько факторов, например, коэффициенты дифференциальных операторов и область, занятая объектом исследования).

Следует отметить, что вышеприведенная классификация обратных задач является весьма условной, и постановки задач легко трансформируются одна в другую.

К сожалению, решение многих обратных задач можно найти лишь приближенно, при помощи численных алгоритмов, и требуются достаточно тонкие математические средства анализа для обоснования сходимости и устойчивости решений таких задач.

Обратные задачи обладают рядом неприятных, с точки зрения обработки информации, свойств. Во-первых, как правило, обратные задачи являются нелинейными. Во-вторых, возможна неединственность при решении обратных задач, и, в-третьих, наиболее неприятным свойством является их неустойчивость по отношению к малым изменениям входной информации.

Для обратных задач погрешность, присущая всем измерениям, может оказывать очень сильное влияние на погрешность восстановления каких-либо свойств объекта. Это означает, что увеличение точности проведения эксперимента не может кардинально улучшить ситуацию в процедуре идентификации. Задачи, обладающие такими свойствами, принято называть некорректными; подобные ситуации являются весьма непривычными в инженерной практике и в технике эксперимента, требуют адекватных математических средств для описания и построения устойчивых вычислительных алгоритмов обработки.

## Описание теоретической установки и процесса измерений

Теоретическая модель установки представляет собой систему, позволяющую генерировать и регистрировать упругие волны, проходящие через пористые материалы[17].

В процессе измерений используется источник упругих волн, который генерирует волны, направленные в пористый материал сверху. Под этим пористым материалом находится еще один пористый материал (полупространство, ограниченное сверху первым материалом). Затем эти волны проходят через материал и регистрируются с помощью приемника, расположенного на некотором расстоянии от источника. Приемник записывает данные о медленностях волн и их частотах, которые являются обратными значениями скоростей и характеристиками волн, соответственно. Данная установка изображена на рисунке 1.

Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание

Рисунок 1 – Модель установки

## Архитектура искусственных нейронных сетей

Искусственная нейронная сеть представляет собой сеть, состоящую из набора нейронов, соединенных с друг другом помощью взвешенных связей.

Математическая модель искусственного нейрона показана на рисунке 2, в которой, и являются входами и синапсами (весами) –го нейрона; Y – выход; – смещение (вес единичного входа). А текущее состояние нейрона определяется, как взвешенная сумма его входов[8]:

И выход нейрона есть функция его состояния:

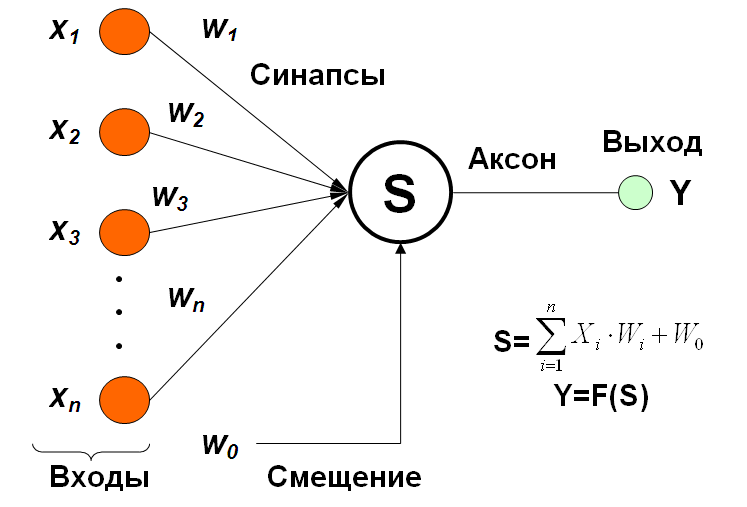


Рисунок 2 – Математическая модель искусственного нейрона

Блок-схема общей математической модели процессора сети приведенной на рисунке 2, представлена на рисунке 3. В этой модели блок 2, обозначенный , является сумматором (соединением входов и их веса – цифра 1 на рисунке), результат которого поступает в блок номер 3, обозначенный , где происходят действия функции активации. В результате получается выходной сигнал Y, обозначенный цифрой 4.

Изображение выглядит как текст, диаграмма, линия, Шрифт

Автоматически созданное описание

Рисунок 3 – Математическая модель нескольких входных нейронов

### Обучение нейронной сети

Существует несколько методов обучения, например: обучения с учителем, обучения без учителя, обучение с подкреплением. В этом исследовании рассматривается для дальнейшего использования метод обучения с учителем.

Сеть обучается путем предоставления ей пар входных образцов и желаемых выходов. Эти пары получены во время сбора данных. Алгоритм использует разницу между выходами, вычисленными в сети, с требуемыми выходами, чтобы они соответствовали весам в сети. Это обычно дается как в задаче приближения функций – данные обучения состоят из входных образцовых пар, и соответствующих результатов, целью является найти функцию, удовлетворяющую всем входных образцам. На рисунке 4 схематично изображено как раз нужное нам обучение.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 4 – Модель обучения с учителем

Данная работа относится к технике обучения по параметрам с учителем[13][14].

В контрольном обучении пары вводно-выводных сигналов используются для обучения сети, таким образом, чтобы выходной сигнал приближался к желательному сигналу системы. Это означает, что в процессе обучения, входные значения обучающих образцов добавляются к сети и по потокам распространенных данных в сети создаются выходные значения сети.

Далее идет процесс сравнения значений, созданных нейронной сетью с желаемыми значениями. В идеальном случае эти два значения одинаковы, тогда и заканчивается процесс обучения. Если отклонение между ними превышает допускаемое значение, то процесс обучения продолжается, чтобы изменить весовую матрицу.

Обучение сети — это процесс непрерывный и повторяемый, и не должен быть прерван пока не найдено значение всех весов, удовлетворяющее условиям, что выходное значение нейронной сети равны желаемому значению. Таким образом, обучение заканчивается при достижении для каждого из обучающих образов значения функции ошибки, не превосходящего некоторого заданного значения, либо после максимально допустимого числа итераций (эпох).

Алгоритмы обучения ИНС основаны на минимизации ошибки, характеризующей отклонения выходного сигнала, что особенно важно в поставленной задаче. Эта минимизация проводится различными методами, такими, как классические градиентные методы, так и с использованием генетических алгоритмов. В этой работе представлен алгоритм обратного распространения ошибки.

### Алгоритм для решения обратных коэффициентных задач с помощью нейросетей

Последовательность действий при обучении нейросети для решения задачи идентификации свойств материала для исследования объекта будет строиться в дальнейшей работе по следующему алгоритму:

1. построение модели исследуемого объекта,
2. моделирование множества исследуемых объектов с различными параметрами проницаемости волн,
3. фильтрация входных данных и их нормализация,
4. обучение нейронной сети с учителем с алгоритмом обратного распространения ошибки по приведенной выше системе,
5. сохранение уже обученной нейросети.

Для использования этой нейросети в решении обратных коэффициентных задач теории упругости будет использован следующий алгоритм:

1. получение информации об исследуемом объекте,
2. фильтрация данных и их нормализация,
3. применение обученной нейросети,
4. получение свойств материала,
5. оценка состояния объекта.

# Теоретические основы нейронных сетей

Конечные элементы и некоторые другие методы позволяют решать обратные коэффициентные задачи теории упругости, но они требуют высокой вычислительной мощности и времени. В этой работе рассматривается применение нейросетевых алгоритмов для решения обратных коэффициентных задач теории упругости. Нейросети для решения подобных задач удобны, так как для получения результатов воздействия на материал приходится анализировать большое количество данных.

## Формирование обучающей и тестовых выборок данных

После процесса измерений полученные данные необходимо обработать и подготовить для обучения нейронной сети.

Для получения данных с экспериментальной установки, была предоставлена программа на языке Fortran, генерирующая значения пористостей для двух материалов в промежутке от 0.01 до 0.35 (объем пор в среде / весь объем среды, а после 0.36 среда начинает разрушаться) и значения точек на оси x и оси y на графике рисунка 5 и рисунка 6 для пары материалов.

Данные точки – зависимости медленностей волны от её частоты.

Изображение выглядит как текст, линия, снимок экрана, диаграмма

Автоматически созданное описание

Рисунок 5 – График зависимости

График зависимости медленности волны от её частоты для пористостей и

Изображение выглядит как График, линия, текст, диаграмма

Автоматически созданное описание

Рисунок 6 – График зависимости

График зависимости медленности волны от её частоты для пористостей и

Для создания нейросети формируются обучающая и тестовая выборки данных. Обучающая выборка представляет собой набор записей о медленностях и частотах волн, полученных в результате экспериментов. Тестовая выборка содержит данные, которые не были использованы в процессе обучения и служит для оценки производительности и обобщающей способности нейронной сети.

Пример выборки, которую даёт программа, изображен на рисунке 7.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 7 – Пример обучающих данных

Первые три строчки – изменяемые пользователем параметры (количество первых пористостей, количество вторых пористостей и число промежуточных частот). Далее написана пара частот, ниже точка по оси х, количество соответствующих ей значений по y и сами эти значения.

Для использования этих данных производилась предварительная обработка данных, включающая их очистку от шумов, а также приведение к соответствующему формату.

## Архитектура нейронных сетей

Архитектура нейронной сети определяет структуру и взаимосвязи между нейронами, которые позволяют модели анализировать входные данные и генерировать соответствующий вывод.

Существует множество различных архитектур нейронных сетей, начиная с классических моделей, таких как однослойные персептроны, и заканчивая современными глубокими нейронными сетями, такими как сверточные нейронные сети и рекуррентные нейронные сети. Каждая архитектура имеет свои особенности и предназначена для решения определенных типов задач.

Однослойные персептроны представляют собой самую простую форму нейронной сети, состоящую из одного слоя нейронов. Они подходят для решения линейно разделимых задач.

Сверточные нейронные сети (СНС) эффективно работают с изображениями и визуальными данными. Они используют операцию свертки для автоматического извлечения признаков из входных данных, что позволяет им распознавать образы и выполнять задачи классификации и сегментации изображений.

Рекуррентные нейронные сети (РНС) обладают способностью моделировать последовательные данные и учитывать контекст. Они имеют обратные связи между нейронами, что позволяет им запоминать информацию о предыдущих состояниях и использовать ее для принятия решений в текущем состоянии. РНС широко применяются в задачах обработки естественного языка, машинного перевода, анализа временных рядов и других задачах с последовательными данными.

Выбор архитектуры нейронной сети основывается на характеристиках задачи и доступных данным. В данной работе применена глубокая нейронная сеть, состоящая из нескольких слоев, включая скрытые, которые помогут модели извлекать сложные зависимости из данных измерений волн.

Нейросеть является многослойным персептроном, схематическое изображение которого находится на рисунке 8.

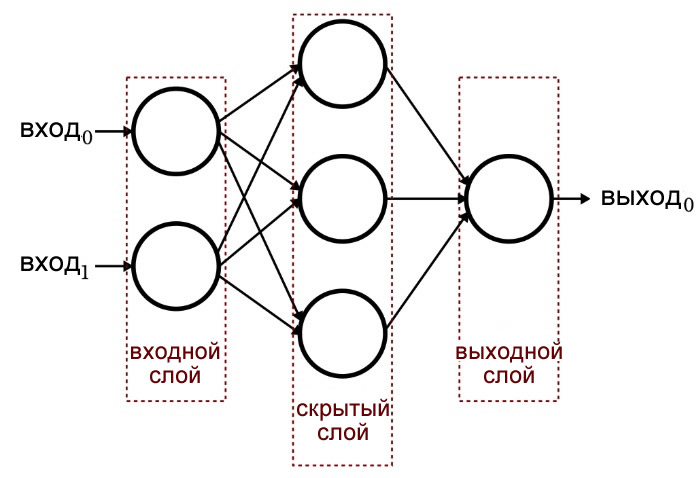


Рисунок 8 – Персептрон

Многослойный персептрон представляет собой классическую архитектуру нейронной сети, состоящую из одного или нескольких скрытых слоев (в данном случае три слоя) между входным и выходным слоями. Каждый слой содержит набор нейронов, которые соединены с нейронами следующего слоя. В данном случае каждый скрытый слой имеет функцию активации ReLU, изображенную на рисунке 9, которая является типичным выбором для персептрона.

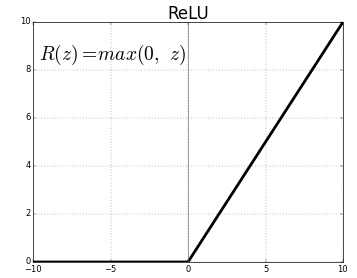


Рисунок 9 – Функция ReLU

Графически, функция ReLU выглядит как линейная функция с отсечением в нуле. Она возвращает входной сигнал, если он положителен, и ноль в противном случае.

В примере метод \_\_init\_\_ инициализирует модель, определяя все слои нейронной сети. Каждый слой представлен объектом класса nn.Linear, который является линейным слоем нейронной сети. В данном случае, модель состоит из четырех линейных слоев, код которых предоставлен на рисунке 10.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Функции с созданием слоёв

Первый линейный слой с входным размером 11 и выходным размером 30. Второй слой линейный слой с входным размером 30 и выходным размером 50. Третий линейный слой с входным размером 50 и выходным размером 10. Выходной линейный слой с входным размером 10 и выходным размером 2.

Метод forward определяет прямой проход модели. Он определяет последовательность операций, которые применяются к входным данным при прохождении через слои нейронной сети.

В данном случае, метод forward принимает входной тензор x (многомерный массив, матрица) и пропускает его через каждый слой с активационной функцией ReLU. После прохождения через каждый слой, результат передается в следующий слой. На последнем слое применяется активационная функция ReLU и полученный результат возвращается в качестве выхода модели.

Таким образом, эти методы инициализируют модель и определяют прямой проход через нее, который выполняется при вызове метода forward.

Нейросеть использует архитектуру с прямым распространением сигнала, состоит из нескольких слоев линейных преобразований и нелинейных функций активации.

В процессе обучения модели проводится 50 эпох обучения, то есть весь обучающий датасет проходит через заданное количество раз. В каждой эпохе происходит проход по обучающему набору данных и формируются батчи – деление датасета на маленькие партии. Для каждого батча вычисляются предсказания модели, рассчитывается значение функции потерь и производится обновление весов модели с использованием обратного распространения ошибки.

## Обратное распространение ошибки

Обратное распространение ошибки является ключевым алгоритмом для обучения нейронных сетей, включая многослойные персептроны. Он позволяет эффективно настраивать веса нейронов сети на основе полученных ошибок предсказания.

Процесс обратного распространения ошибки состоит из следующих шагов:

Прямой проход: входные данные подаются на вход сети, и значения проходят через все слои от входного до выходного. Каждый нейрон в слое получает взвешенную сумму входных значений (сами входные значения, помноженные на веса), применяет к ней функцию активации и передает результат на следующий слой. Проход продолжается до выходного слоя, где генерируются предсказания сети.

Вычисление функции потерь: сравнивается предсказанный результат с желаемым выходом (целевыми значениями) и вычисляется ошибка или функция потерь, которая показывает, насколько сильно предсказания отличаются от желаемого результата.

В данной нейронной сети используется функция потерь – среднеквадратичная ошибка. Она является одной из наиболее распространенных для задач регрессии, когда требуется предсказать непрерывные числовые значения.

Обратное распространение ошибки: ошибка с выходного слоя распространяется обратно через сеть. Каждый нейрон в слое получает информацию об ошибке и вкладе в нее от нейронов следующего слоя. Затем происходит вычисление градиента функции потерь по весам и нейронов.

Обновление весов: после вычисления градиента функции потерь по весам и смещениям, происходит обновление этих параметров. Обычно применяется градиентный спуск или его модификации, чтобы минимизировать функцию потерь. Веса и смещения корректируются в направлении, противоположном градиенту, чтобы уменьшить ошибку предсказания.

Обратное распространение ошибки позволяет эффективно настраивать веса нейронной сети, обучая ее на обучающем наборе данных и улучшая ее способность к предсказанию целевых значений. Этот процесс повторяется множество раз до достижения оптимальных весов, которые минимизируют функцию потерь и обеспечивают наилучшую производительность модели. Данная модель обратного распространения ошибки изображена на рисунке 11.

Изображение выглядит как диаграмма, линия, круг, рисунок

Автоматически созданное описание

Рисунок 11 – Обратное распространение ошибки

На примере созданной нейронной сети и вывода её промежуточных ошибок можно отследить работу алгоритма обратного распространения ошибки. В самом начале ошибка достаточно большая, а далее начинает уменьшаться, что показано на рисунке 12

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 12 – Обратное распространение ошибки на примере вывода кода

В конце алгоритм находит минимальную ошибку, делает ещё одну итерацию, сравнивает предыдущую ошибку и нынешнюю. И так как предыдущая меньше, то алгоритм выбирает для работы эту ошибку и приостанавливает работу, так как минимальная ошибка найдена. Минимальная ошибка и её поиск изображены на рисунке 13.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – Обратное распространение ошибки на примере вывода кода

### Линейная регрессия

В качестве обоснования выбора данного алгоритма обучения можно провести сравнение его эффективности с алгоритмом линейной регрессии.

Модель линейной регрессии – это статистическая модель, которая используется для предсказания непрерывных числовых значений на основе линейной зависимости между входными признаками и целевой переменной. Она представляет собой простую и интерпретируемую модель, которая находит линейную связь между признаками и целевой переменной. Алгоритм линейной регрессии предоставлен на рисунке 14.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 14 – Создание метода линейной регрессии

Можно сделать вывод, что раз ошибка значительно больше, то модель для реализуемой нейронной сети выбрана верно.

## Классификация задач для нейронной сети

Задачи для нейронной сети можно разделить на два вида: задачи классификации и регрессии.

Задачи классификации:

В задачах классификации нейронные сети используются для прогнозирования категориальных (дискретных) меток или классов на основе входных данных. Например, классификация изображений на категории, определение эмоционального тона текста, определение спама в электронной почте и другие подобные задачи. В таких задачах нейронная сеть обучается на наборе данных с известными метками, чтобы выявить закономерности и обучиться классифицировать новые примеры.

Задачи регрессии:

В задачах регрессии нейронные сети используются для предсказания непрерывных числовых значений на основе входных данных. Например, предсказание цены недвижимости на основе ее характеристик, предсказание объема продаж в зависимости от рекламных затрат, предсказание времени выполнения задачи и другие подобные задачи. Нейронная сеть обучается на наборе данных с известными числовыми значениями, чтобы выявить зависимости и настроиться на предсказание новых значений.

В задачах классификации обычно используются функции активации, которые помогают модели предсказывать вероятности принадлежности к каждому классу, например, softmax.

Softmax – функция, превращающая наборы чисел в вероятности, чья сумма равна единице. Функция выводит в качестве результата вектор, представляющий распределения вероятностей списка потенциальных результатов. В задачах регрессии функция активации может быть линейной или нелинейной, в зависимости от природы данных.

## Сравнение результатов при изменении параметров

На качество обучения влияет не только архитектура сети, но и качественно подобранное сочетание параметров данных обучения. Для обоснования оптимальности выбора данных для обучения, можно рассмотреть данную таблицу 1.

Таблица 1 – Сравнение ошибки при разных данных для обучения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Количество пористостей | Количество частот | Соотношение текстовой к обучающей выборки | Минимальная ошибка |
| 1 | 10 | 30 | 33 | 172.0114 |
| 2 | 10 | 30 | 20 | 104.4146 |
| 3 | 10 | 60 | 33 | 362.7562 |
| 4 | 10 | 60 | 20 | 183.9152 |
| 5 | 10 | 100 | 33 | 101.0729 |
| 6 | 10 | 100 | 20 | 89.8869 |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 10 | 130 | 33 | 252.0384 |
| 8 | 10 | 130 | 20 | 192.2895 |
| 9 | 20 | 30 | 33 | 60.7559 |
| 10 | 20 | 30 | 20 | 56.5915 |
| 11 | 20 | 60 | 33 | 69.9528 |
| 12 | 20 | 60 | 20 | 314.1696 |
| 13 | 20 | 100 | 33 | 82. 6206 |
| 14 | 20 | 100 | 20 | 373.0068 |

Как правило, обучающая выборка составляет 75 – 80% от объема исходных данных, хотя каких-то строгих правил в этом отношении не существует, поэтому в пределах допустимых значений процент этой выборки колебался. Так же изменялись количество пропускаемых частот и количество пар пористостей в пределах от 0.01 до 0.35.

Исходя из проанализированных данных должна быть выбрана комбинация номер 10, однако такое количество данных пар точек может оказаться не показательным для результата. Поэтому была использована в работе комбинация номер 13, как комбинация с оптимальным набором данных и минимальной ошибкой.

# ЗАКЛЮЧЕНИЕ

В данной курсовой работе были разработаны и применены методы анализа данных упругих волн с использованием нейронных сетей. Нейронная сеть была обучена на экспериментальных данных, полученных из установки с двумя пористыми материалами. По входным параметрам – медленностям волн и частотам, нейронная сеть предсказывает пористость первого и второго материалов. Полученные результаты показывают, что нейронная сеть способна эффективно решать обратные коэффициентные задачи теории упругости.

Подводя итоги исследования, можно отметить успешность применения нейронных сетей в решении обратных коэффициентных задач теории упругости. Нейронная сеть показала способность предсказывать пористость материалов на основе медленностей волн и частот, что является важным шагом в анализе и определении свойств пористых материалов.

В ходе работы были достигнуты поставленные цели и выполнены задачи исследования. Был разработан алгоритм работы нейронной сети, проведена обработка и анализ экспериментальных данных, а также оценена эффективность нейронной сети для решения обратных коэффициентных задач.

Исследование имеет практическую значимость в контексте разработки методов и инструментов для определения пористости материалов на основе данных упругих волн. Полученные результаты могут быть полезны в различных областях, где важно определить структурные характеристики пористых материалов, таких как геология, строительство, нефтегазовая промышленность и другие.

Необходимо отметить некоторые ограничения, с которыми столкнулась данная работа. Во-первых, использовались данные только с одной установки, что может ограничивать обобщение результатов на другие системы. Для более широкого применения и достоверности результатов рекомендуется расширение объема экспериментальных данных и анализ данных с нескольких установок.

Дальнейшие исследования могут быть направлены на улучшение архитектуры и параметров нейронной сети, а также на использование других методов машинного обучения для решения обратных коэффициентных задач. Также можно исследовать возможность расширения модели для учета других физических свойств пористых материалов или комбинировать данные различных типов для повышения точности предсказаний.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Назначение и виды неразрушающего контроля [электронный ресурс]. – URL: <https://www.omgtu.ru/general_information/faculties/radio_engineering_department/department_quot_radio_devices_and_diagnostic_systems_quot/educational-materials/Nondestructive_testing/Lecture_notes_2015.pdf> (дата обращения 7.05.2023)
2. Glushkov, E.V., Glushkova, N.V., Eremin, A.A., Evdokimov, A.A., Lammering, R. (2016). Ultrasonic Guided Wave Characterization and Inspection of Laminate Fiber-Reinforced Composite Plates. In: Parinov, I., Chang, SH., Topolov, V. (eds) Advanced Materials. Springer Proceedings in Physics, vol 175. Springer, Cham.
3. Golub, M.V.; Doroshenko, O.V.; Arsenov, M.A.; Bareiko, I.A.; Eremin, A.A. Identification of Material Properties of Elastic Plate Using Guided Waves Based on the Matrix Pencil Method and Laser Doppler Vibrometry. Symmetry 2022, 14, 1077.
4. Solving differential equations with neural networks: Applications to the calculation of cosmological phase transitions (Решение дифференциальных уравнений с помощью нейронных сетей: приложения к расчету космологических фазовых переходов) [электронный ресурс]. – URL: <https://arxiv.org/abs/1902.05563> (дата обращения 10.05.2023)
5. Using Neural Networks to solve Ordinary Differential Equations (Использование нейросетей для решения обычных дифференциальных уравнений) [электронный ресурс]. – URL: <https://towardsdatascience.com/using-neural-networks-to-solve-ordinary-differential-equations-a7806de99cdd> (дата обращения 12.05.2023)
6. Седов Л. И. Механика сплошной среды. Том 1. М.: Наука/ Л. И. Седов – 1971 г. – 492 с. – ISBN 9789001796808
7. Tariq Rashid Make Your Own Neural Network 1st Edition/ Tariq Rashid – reateSpace Independent Publishing Platform; 1st edition – 2016. – 222 с. – ISBN 978-1530826605
8. Ватульян А. О. Обратные задачи в механике деформируемого твердого тела/ А. О. Ватульян – Физматлит М. – 2007. – 224 с. – ISBN 9785922108355
9. Новацки В. Теория упругости / В. Новацки – Мир. – 1975. – 864 с
10. Парадигмы обучения нейронных сетей. Алгоритм решения задач с помощью нейронных сетей. Алгоритм обратного распространения ошибки. Расписание обучения нейронных сетей [электронный ресурс]. – URL: [http://apsheronsk.bozo.ru/Neural/Lec3.htm](http://apsheronsk.bozo.ru/Neural/Lec3.htm%20) (дата обращения 10.05.2023)
11. Нейросетевой подход к решению Коэффициентной обратной задачи [электронный ресурс]. – URL: <https://pandia.ru/text/78/299/971.php> (дата обращения 10.05.2023)
12. Гафаров Ф.М Искусственные нейронные сети и приложения: учеб. пособие/ Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с. –УДК 004.032.26
13. Обучение нейросети с учителем, без учителя, с подкреплением [электронный ресурс]. – URL: [https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem](https://neurohive.io/ru/osnovy-data-science/obuchenie-s-uchitelem-bez-uchitelja-s-podkrepleniem%20) (дата обращения 23.04.2023)
14. Пример пошагового выполнения алгоритма обратного распространения ошибки [электронный ресурс]. – URL: [http://www.habarov.spb.ru/lab\_nnet/nn\_js\_lab/content/nn\_js\_backprop\_2.html](http://www.habarov.spb.ru/lab_nnet/nn_js_lab/content/nn_js_backprop_2.html%20) (дата обращения 11.05.2023)
15. Giuseppe Ciaburro Keras 2.x Projects/ Giuseppe Ciaburro Packt Publishing – 2018. – 394 с. – ISBN 9781789534160
16. Глушков, Е. В., Глушкова Н. В., Фоменко С. И. Влияние пористости на Характеристики волн рэлеевского типа в многослойном полупространстве / Е. В. Глушков, Н. В. Глушкова, С. И. Фоменко – 2011 г. – 12 с.

# ПРИЛОЖЕНИЕ А

Код программы

import torch

import torch.nn as nn

from torch.nn import functional as F

from sklearn.model\_selection import train\_test\_split

import numpy as np

from sklearn.preprocessing import StandardScaler

class Dataset(torch.utils.data.Dataset):

def \_\_init\_\_(self, X, y):

super().\_\_init\_\_()

self.X = X

self.y = y

def \_\_len\_\_(self):

return len(self.X)

def \_\_getitem\_\_(self, idx):

return torch.Tensor(self.X[idx]), torch.Tensor(self.y[idx])

with open("DispMatr.txt", "r") as f:

lines = f.readlines()[3:]

data = []

for i in range(0, len(lines), 101):

lines\_t = lines[i: i + 101]

por1, por2 = (float(val) for val in lines\_t[0].split())

for j in range(1, len(lines\_t)):

points\_t = [float(val) for val in lines\_t[j].split()]

n\_y = points\_t.pop(1)

points\_t += [0 for i in range(10 - int(n\_y))] + [por1, por2]

data.append(points\_t)

class Model(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.dense1 = nn.Linear(11, 30)

self.dense2 = nn.Linear(30, 50)

self.dense3 = nn.Linear(50, 10)

self.dense\_out = nn.Linear(10, 2)

def forward(self, x):

x = F.relu(self.dense1(x))

x = F.relu(self.dense2(x))

x = F.relu(self.dense3(x))

x = F.relu(self.dense\_out(x))

return x

device = "cpu"

model = Model().to(device)

criterion = nn.MSELoss()

optimizer = torch.optim.SGD(model.parameters(), lr=0.1, momentum=0.9)

EPOCHS = 50

X, y = np.array(data)[:, :-2], np.array(data)[:, -2:]

X = StandardScaler().fit\_transform(X)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.33, random\_state=42)

train\_dataset = Dataset(X\_train, y\_train)

val\_dataset = Dataset(X\_test, y\_test)

train\_dataloader = torch.utils.data.DataLoader(train\_dataset, batch\_size=10, shuffle=True)

val\_dataloader = torch.utils.data.DataLoader(val\_dataset, batch\_size=64, shuffle=True)

from turtle import clear

from tqdm import tqdm

from IPython.display import clear\_output

def get\_accuracy(labels, preds):

return abs((labels - preds) / labels[0]).sum() / labels.shape[0] \* 100

max\_acc = 100000

for epoch in range(EPOCHS):

running\_loss = 0.0

running\_acc = 0.0

for i, (inputs, labels) in tqdm(enumerate(train\_dataloader)):

optimizer.zero\_grad()

outputs = model(inputs.to(device))

loss = criterion(

outputs, labels.to(torch.float32).to(device) # Считаем loss

)

loss.backward()

optimizer.step()

running\_loss += loss.item()

running\_acc += get\_accuracy(labels.to(device), outputs)

print(f"[train epoch {epoch + 1}] loss: {running\_loss / i:.3f} MAPE: {running\_acc / i:.3f}")

val\_loss = 0.0

val\_acc = 0.0

for i, (inputs, labels) in enumerate(val\_dataloader):

optimizer.zero\_grad()

with torch.no\_grad():

outputs = model(inputs.to(device))

loss = criterion(outputs, labels.to(torch.float32).to(device))

val\_loss += loss.item()

val\_acc += get\_accuracy(labels.to(device), outputs)

print(

f"[val epoch {epoch + 1}] loss: {val\_loss / len(val\_dataloader):.3f} MAPE: {val\_acc / len(val\_dataloader):.3f}"

)

if val\_acc.item() / len(val\_dataloader) < max\_acc:

max\_acc = val\_acc / len(val\_dataloader)

torch.save(model.state\_dict(), f"models/model\_MAPE\_{max\_acc}.pt")

print("min error:", max\_acc)

from sklearn.linear\_model import LinearRegression as LR

lr = LR().fit(X\_train, y\_train)

pred = lr.predict(X\_test)

get\_accuracy(y\_test, pred)