ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему: Нейросети с нуля

(промежуточный, этап 1)

Выполнил:
Студент группы БКНАД221 _____ Иван Игоревич Плешков
Подпись И.О.Фамилия

06.02.2024
Дата

Принял:
Руководитель проекта Дмитрий Витальевич Трушин
Имя, Отчество, Фамилия

доцент, к.ф.-м.н.
Должность, ученое звание

ФКН НИУ ВШЭ
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки 06.02 2024 _____ _____
Оценка (по 10-ти бальной шкале) Подпись

Москва 2024

Содержание

Аннотация

В рамках данного курсового проекта рассматривается разработка и реализация нейронной сети на языке программирования C++ для задачи распознавания рукописных цифр. Проект направлен на глубокое изучение теоретических основ нейронных сетей, а также практическое применение полученных знаний для создания работающей модели.
Ключевые слова: машинное обучение, нейронные сети, распознавание цифр, обратное распространение ошибки, градиентный спуск.

1 Введение

Нейронные сети представляют собой мощные вычислительные системы, вдохновленные структурой и принципами работы человеческого мозга. Благодаря их способности к обучению нейронные сети находят широкое применение в различных областях, от распознавания образов, звука и текста до прогнозирования временных рядов и автономного управления транспортными средствами. Нейронные сети состоят из взаимосвязанных слоев "нейронов"– вычислительных единиц, которые обрабатывают входные данные и передают результаты дальше по сети. Каждый отдельный нейрон можно представлять как функцию, принимающую на вход несколько значений и выдающую один результат. Нейронные сети работают эффективно, потому что они умеют обучаться, используя различные алгоритмы обучения. Одним из основных методов обучения нейронных сетей является метод обратного распространения ошибки. Во время обучения этот алгоритм работает в обратном направлении, сравнивая выходные данные каждого слоя с ожидаемыми результатами, оценивая ошибки и корректируя параметры слоев, чтобы улучшить работу сети. Для людей, не знакомых с принципами работы нейронных сетей, структура слоев нейронов и метод обратного распространения ошибки могут представлять собой своеобразный "черный ящик".

Цель данного курсового проекта – изучить устройство нейронных сетей, методов их обучения и разработать собственную нейронную сеть с нуля на языке программирования C++ для распознавания рукописных цифр. Это не просто техническая задача реализации архитектуры нейросети, но и глубокое погружение в основы теории нейронных сетей, изучение их математической модели, алгоритмов обучения и принципов работы. Проект предполагает не только теоретический анализ существующих методов, но и практическое применение этих знаний для создания работоспособной модели, способной эффективно распознавать рукописные цифры на основе датасета MNIST.

Основными задачами проекта являются:

- Изучение теоретических основ нейронных сетей, включая их структуру, принципы обучения и алгоритмы оптимизации.
- Разработка и реализация компонентов нейронной сети, включая слои, функций активаций, функции потерь и общую архитектуру сети.
- Обучение разработанной модели на датасете MNIST и оценка её способности корректно распознавать рукописные цифры.
- Проведение различных тестов, сравнение и анализ разных конфигурации нейронной сети.

2 Функциональные требования

Данный проект представляет собой программу для распознавания рукописных цифр, разрабатываемую на языке программирования C++17. Для реализации вычислений, связанных с линейной алгеброй, используется библиотека Eigen, что позволяет эффективно работать с матрицами и векторами, необходимыми для обработки данных и обучения сети. Контроль версий осуществляется с помощью системы git. Для облачного хранения и публикации исходного кода используется платформа GitHub. На данный момент написаны ключевые классы для работы нейронной сети. Далее описаны классы, которые были реализованы на данный момент в рамках проекта, и их ключевые функциональные аспекты.

2.1 Класс `NeuralNetwork`

Класс, интегрирующий все необходимые компоненты в единую структуру нейронной сети. Он отвечает за:

- Сборку сети из последовательности слоев с заданными параметрами.

- Выполнение прямого распространения входного сигнала через слои сети для получения предсказания.
- Выполнение обратного распространения ошибки для обновления весов сети на основе заданной функции потерь и скорости обучения.
- Управление процессом обучения сети, включая подачу обучающих примеров и адаптацию параметров сети для минимизации ошибки.

2.2 Класс `Layer`

Этот класс представляет собой один слой нейронной сети, состоящий из нейронов. Основные функции класса включают:

- Инициализацию весов и смещений нейронов слоя с использованием нормального распределения для начальных значений.
- Выполнение прямого распространения, принимая входные данные и применяя активационную функцию для получения выходных значений слоя.
- Выполнение обратного распространения, обновляя веса и смещения на основе градиента потерь.
- Обновление весов в соответствии с полученным градиентом и заданной скоростью обучения.

2.3 Класс `ActivationFunction`

Абстрактный базовый класс для активационных функций, определяющий интерфейс для вычисления активационной функции и её производной. Каждая конкретная функция активации должна реализовать следующие методы:

- `compute`: Вычисляет значение функции активации для входа `x`.
- `computeDerivative`: Вычисляет производную функции активации в точке `x`.

2.4 Класс `SigmoidActivation`

Реализация класса `ActivationFunction`, представляющая сигмоидальную функцию активации. Сигмоидальная функция характеризуется гладким переходом выходного сигнала от 0 к 1, что делает её подходящей для использования в последнем слое нейронной сети при решении задач классификации.

2.5 Класс `LossFunction`

Абстрактный класс, описывающий интерфейс для функций потерь. Функция потерь необходима для оценки эффективности сети на этапе обучения. Ключевые методы класса включают:

- `computeLoss`: Вычисляет значение функции потерь, сравнивая предсказанные значения с фактическими.
- `computeDerivativeLoss`: Вычисляет градиент функции потерь, необходимый для обратного распространения ошибки.

2.6 Класс `MeanSquaredError`

Конкретная реализация `LossFunction`, представляющая функцию среднеквадратичной ошибки (MSE). MSE является стандартной метрикой для оценки производительности сети в задачах регрессии и классификации, где необходимо минимизировать разницу между предсказанными и реальными значениями.

3 Взаимодействие классов в архитектуре нейронной сети

В архитектуре разработанной нейронной сети ключевыми элементами являются классы `NeuralNetwork`, `Layer`, `ActivationFunction`, включая его конкретные реализации, и классы функций потерь. Взаимодействие этих классов обеспечивает полный цикл работы нейросети, от прямого распространения сигнала до обучения сети с использованием обратного распространения ошибки.

Сборка и инициализация сети. Класс `NeuralNetwork` отвечает за создание и инициализацию нейронной сети, агрегируя в себе множество слоёв (`Layer`). При инициализации экземпляра `NeuralNetwork`, он последовательно создаёт слои сети, каждый из которых инициализируется с заданным количеством нейронов и выбранной функцией активации, например, `SigmoidActivation`.

Прямое распространение. В процессе прямого распространения, класс `Layer` принимает входные данные, выполняет матричное умножение входных данных на матрицу весов слоя, добавляет смещение и применяет функцию активации. Результатом является выходной вектор слоя, который передаётся как вход следующему слою в сети.

Обратное распространение и обновление весов. После получения значения функции потерь, например, с использованием `MeanSquaredError`, начинается процесс обратного распространения ошибки. Каждый слой сети (`Layer`) обновляет свои веса и смещения в соответствии с градиентом функции потерь, что позволяет минимизировать ошибку на выходе сети. Обновление параметров слоя осуществляется на основе градиентов, полученных в ходе обратного распространения, и скорости обучения.

Функции активации и потерь. Функция активации, на данный момент реализованная в классе `SigmoidActivation`, применяется к каждому нейрону слоя для внесения нелинейности в процесс обработки данных сетью. Функции потерь, на данный момент реализована `MeanSquaredError`, оценивают разницу между предсказаниями сети и реальными данными, что критически важно для процесса обучения.

4 Нефункциональные требования

- C++17
- Библиотека Eigen для работы с линейной алгеброй
- Система контроля версий Git
- GitHub – платформа для хранения и управления программным обеспечением с использованием системы контроля версий Git.

5 Обзор литературы

Для реализации проекта была изучена документация языка C++17[1]. Также для написания более качественного кода была прочитаны книга "C++ Primer Plus"[2]. Для работы с линейной алгеброй была изучена документация библиотеки Eigen[3], а также документация системы контроля версий Git[4]. Теория, необходимая для реализации нейронной сети, была изучена в книгах "Математические основы машинного обучения и прогнозирования"[5] и "Neural Networks and Deep Learning"[6], а также в курсе Евгения Соколова "Машинное обучение на ФКН ВШЭ"[7]. Для изучения различных алгоритмов градиентного спуска была изучена статья "An overview of gradient descent optimization algorithms"[8], а также две статьи про алгоритм оптимизации Adam[9] и алгоритм SAG[10].

6 План дальнейшей работы

В рамках дальнейшей работы над проектом запланированы следующие ключевые этапы работы:

1. Доработка и добавление классов необходимых для полноценной работы нейронной сети, включая слои с различными типами активационных функций и методы оптимизации для обучения сети.
2. Написание тестов на работоспособность кода для обеспечения надежности и стабильности работы всех компонентов нейросети. Тестирование позволит своевременно выявлять ошибки и недоработки в логике и алгоритмах.

3. Прохождение код ревью от научного руководителя для выявления потенциальных ошибок и улучшения качества кода. Это поможет обеспечить соответствие кода лучшим практикам разработки и улучшить его читаемость и поддерживаемость.

4. Доработка кода после прохождения код ревью, включая исправление выявленных недочетов и реализацию рекомендованных улучшений.

5. Обучение сети на датасете MNIST и оценка её эффективности работы. В этом этапе будет произведена настройка гиперпараметров сети и выбор наиболее эффективной архитектуры для задачи распознавания рукописных цифр.

6. Проведение различных экспериментов с использованием различной конфигурации нейронной сети, включая эксперименты с разными функциями потерь и функциями активаций, для выявления оптимальных параметров сети и улучшения качества распознавания.

7. Анализ результатов экспериментов для определения наилучших практик и подходов в построении и обучении нейросетей на примере задачи распознавания рукописных цифр.

8. Оптимизация производительности сети, включая исследование и применение техник уменьшения времени обучения и увеличения скорости предсказания.

Список литературы

[1] C++ reference. https://en.cppreference.com/w/cpp/17. (дата обр. 2023-11-23).

[2] Stephen Prata. *C++ Primer Plus*. Addison-Wesley Professional, 6 edition, 2011.

[3] Eigen documentation. https://eigen.tuxfamily.org/index.php?title=Main_Page#Documentation. (дата обр. 2023-12-18).

[4] Git documentation. https://git-scm.com/doc. (дата обр. 2023-11-06).

[5] Вьюгин В.В. *Математические основы теории машинного обучения и прогнозирования*. МЦНМО, 2013.

[6] Michael Nielsen. Neural networks and deep learning. http://neuralnetworksanddeeplearning.com/index.html. (дата обр. 2024-01-20).

[7] Е. А. Соколов. Курс "Машинное обучение"на ФКН ВШЭ. https://github.com/esokolov/ml-course-hse/tree/master?tab=readme-ov-file. (дата обр. 2024-01-10).

[8] Sebastian Ruder. An overview of gradient descent optimization algorithms. 2016.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2017.

[10] Nicolas Le Roux Mark Schmidt and Francis Bach. Minimizing finite sums with the stochastic average gradient. 2016.