Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Южно-Уральский государственный университет»

* 1. (национальный исследовательский университет)

Институт естественных и точных наук

Кафедра прикладной математики и программирования

Отчет по лабораторной работе № 1

по дисциплине «Современные нейросетевые технологии»

|  |  |
| --- | --- |
|  |  |
|  | Авторы работы  Студент группы ЕТ-122  \_\_\_\_\_\_\_\_\_/ Плешков М. С.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. |
|  |  |
|  | Руководитель работы,  \_\_\_\_\_\_\_\_\_\_/ Кичеев Д. М.  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2025 г. |

Челябинск 2025

# Оглавление

1 задание 3

2 теория 4

3 программная реализация 7

4 руководство по сборке и запуску 10

# Задание

Цель: изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой).

Задачи: Выполнение практической работы предполагает решение следующих задач:

1. Изучение общей схемы метода обратного распространения ошибки.

2. Вывод математических формул для вычисления градиентов функции ошибки по параметрам нейронной сети и формул коррекции весов.

3. Проектирование и разработка программной реализации.

4. Тестирование разработанной программной реализации.

5. Подготовка отчета, содержащего минимальный объем информации по каждому этапу выполнения работы.

В процессе выполнения лабораторной работы предполагается, что сеть ориентирована на решение задачи классификации одноканальных изображений. Типичным примером такой задачи является задача классификации рукописных цифр. Именно ее предлагается использовать в качестве тестовой задачи на примере набора данных MNIST [1].

Метод обратного распространения ошибки разрабатывается, исходя из следующих предположений:

1. На входе сети имеется 𝑤×ℎ нейронов, что соответствует разрешению изображения.

2. На выходе сети имеется 𝑘 нейронов, что соответствует количеству классов изображений.

3. Скрытый слой содержит 𝑠 нейронов.

4. В качестве функции активации на втором слое используется функция softmax.

# теория

Для решения задачи классификации цифр из MNIST была разработана программа, реализующая двухслойную нейронную сеть, имеющую заданную структуру:

1. Входной слой размерностью 784.
2. Полносвязный переход 784 на h (кол-во нейронов скрытого слоя).
3. Функция активации SiLU.
4. Полносвязный переход h на 10.
5. Функция активации Softmax.
6. Функция потерь Crossentropy.

Полносвязный переход: , где A – матрица весов слоя, b – вектор весов смещения слоя; y – исходящий вектор, x – входящий вектор.

где y – выходные значения сети, z – ожидаемые выходные значения сети.

Для выполнения обучения сети необходимо для каждого примера выполнять прямой и обратный проходы. Выполнение прямого прохода и вычисление функции потерь:

где – входящий вектор, – результирующий вектор, – значение функции потерь, z – вектор ожидаемых значений.

Далее нужно найти значение функции потерь и вычислить градиент функции потерь по значению веса (градиент – направление наискорейшего роста), чтобы выполнить шаг обучения.

В частности, для вычисления значения градиента по весам для перехода от скрытого слоя к выходному нужно вычислить

Для последнего градиента всё просто

С остальными всё немного интереснее

Отсюда имеем:

Поскольку, исходя из задачи (классификация)

Также нам понадобится производная функции SiLU

Теперь вычислим градиенты обучаемых весов в ходе обратного прохода

Для выполнения самого шага обучения необходимо отнять (поскольку градиент указывает направление наискорейшего роста функции потерь, а нас интересует наискорейшее падение) от обучаемых весов их градиент, умноженный на скорость обучения .

Также может быть полезным знать метрику Accuracy

# программная реализация

Была написана программная реализация, включающая следующие сущности:

1. Класс NN2 двухслойной нейронной сети с функциями активации SiLU и Softmax.
2. Функция main, обеспечивающая приём параметров и запуск программы в соответствующем им режиме.
3. Функция ReadCSV, обеспечивающая считывание данных для работы.

Класс NN2 является шаблонным классом с параметрами w и h для задания размеров входного слоя, h – размер скрытого слоя и k – размер выходного слоя

Класс содержит следующие поля (все – матрицы и вектора действительных чисел одинарной точности):

1. Матрицы обучаемых весов перехода layer1 и layer2.
2. Вектора обучаемых весов смещения bias1 и bias2.

Класс содержит следующие методы:

1. Конструктор без аргументов, не инициализирует веса.
2. Неявный деструктор.
3. init\_rand – используется для инициализации весов случайными значениями от -1 до 1, не принимает параметров.
4. sigmoid – функция активации, используемая для SiLU, static, принимает входящий вектор, возвращает вектор значений функции.
5. silu – функция активации скрытого слоя классификатора, static, принимает входящий вектор, возвращает вектор значений функции.
6. silu\_d – производная функции активации скрытого слоя классификатора, static, принимает входящий вектор, возвращает вектор значений производной функции в точке.
7. softmax – функция активации выходного слоя классификатора, static, принимает ссылку на вектор для операции по месту, не возвращает значений.
8. crossentropy\_loss – функция потерь (используется только для вывода в консоль, так как её производная частично сокращается с производной функции softmax), static, принимает входящий вектор, возвращает одно значение в точке.
9. forward – выполняет прямой проход, принимает вектор входящих значений, возвращает вектор значений выходного слоя.
10. forward\_backward – выполняет прямой проход, обратный проход и корректировку обучаемых весов, принимает ссылку для записи значений выходного слоя, вектор входящих значений, вектор ожидаемых значений выходного слоя, скорость обучения, возвращает значение функции потерь.
11. vec2class – принимает вектор значений выходного слоя, возвращает номер класса с наибольшей степенью уверенности, static.
12. predict – совмещает forward и vec2class.
13. train – выполняет эпоху обучения (forward\_backward для всех задач из датасета), вычисляет среднее значение функции потерь и значение метрики accuracy, принимает массивы входящих и ожидаемых выходящих векторов обучающего датасета и скорость обучения, печатает в консоль метрики, возвращает среднее значение функции потерь.
14. validate\_score – вычисляет среднее значение функции потерь и значение метрики accuracy для тестового датасета, принимает массивы входящих и ожидаемых выходящих векторов тестового датасета, печатает в консоль метрики, возвращает среднее значение функции потерь.
15. save\_weights – выполняет запись параметров сети в бинарный файл, принимает имя файла, кидает исключения при ошибке записи.
16. load\_weights – выполняет загрузку параметров из бинарного файла в объект сети, кидает исключения при ошибке чтения, открытия файла или несовпадении шаблонных параметров или контрольной суммы.

Таблица 1 – Формат бинарного файла

|  |  |
| --- | --- |
| 0 – 4 | “NN2\0” |
| 4 – 16 | 3 размера слоёв |
| 16 – (size-4) | Дамп обучаемых параметров |
| (size-4) – size | Crc32(0–(size-4)) |

Формат CSV датасета: Разделитель элементов «,», разделитель задач – перенос строки. Каждая задача состоит из номера класса как первый элемент (для датасета с неизвестным классом нужно применять любой валидный номер класса), остальные элементы являются значениями входного слоя от 0 до 255.

Для тестового набора MNIST, h=800, epochs=100, lr=1e-4 получается следующий результат (вывод значений по эпохам представлен в репозитории в файле “test\_run.txt”)

Result: loss: 0.57757 accuracy: 0.949

# руководство по сборке и запуску

Исходный код проекта можно скачать с репозитория GitHub   
<https://github.com/pl752/MNNT-1.git>

Проект имеет следующие зависимости:

1. Компилятор с поддержкой C++14.
2. Библиотека Eigen 3 (header-only).
3. Библиотека Boost 1.87 с собранным модулем program\_options.

Предусмотрено несколько вариантов сборки:

1. Visual Studio Build Tools:

Нужно собрать проект vcxproj в release режиме (иначе очень медленно работает), обратите внимание, что нужно установить nuget пакеты, перечисленные в packages.config, но это должно произойти автоматически.

Рекомендуется включить поддержку AVX инструкций, если они имеются (свойства проекта – C/C++ – создание кода – включить расширенный набор инструкций).

1. Meson:

Требуется установить meson, cmake и ninja, а также компилятор для вашей среды.

Выполнить команды

meson setup build

cd build

meson compile

Зависимости должны установиться и настроиться автоматически

1. Ручная сборка:

Нужно скачать Eigen и Boost, для Boost выполнить

bootstrap.bat (или sh) (в среде сборки)

b2 --with-program\_options

В папке stage будут собранные модули

Затем скомпилировать и пролинковать файл main.cpp с указанием директорий включения и библиотек, на Unix-подобных системах boost может иметь динамические библиотеки.

Для получения списка опций и сценариев запуска запустите программу с флагом --help (аргументы в Visual studio можно задать в свойствах проекта Отладка – Аргументы команды).

Сценарии запуска

1. Обучение сети

program train <train>.csv <validate>.csv <weights\_out>.bin [options]

Принимает пути до обучающего датасета, тестового датасета и пути для записи файла весов

Принимает пути до файла с тестовым датасетом и файла весов, печатает среднее значение функции потерь и accuracy для каждой эпохи

Опции:

--epochs – количество эпох (100 по-умолчанию)

--lr – скорость обучения (1e-4 по-умолчанию)

--seed – сид генератора случайных чисел (time(0) по-умолчанию)

Например

MNNT-1.exe train example/mnist\_train.csv example/mnist\_test.csv model.bin --lr 5e-5 --epochs 200

1. Тест сети

program test <test>.csv <weights>.bin [options]

Принимает пути до файла с тестовым датасетом и файла весов, печатает среднее значение функции потерь и accuracy.

Опций нет.

1. Выполнить оценку классов цифр в датасете и вывести в текстовый файл или консоль

program infer <numbers>.csv <weights>.bin <results>.txt/--cout [options]

Принимает пути до файла с тестовым датасетом и файла весов, а также либо флаг --cout для печати в консоль или путь до текстового файла, выводит по одному номеру класса в строке.

Во входном файле для каждой задачи номер класса должен быть указан, но не используется и должен быть любым от 0 до 9.