

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

Факультет компьютерных наук  
Образовательная программа «Прикладная математика и информатика»

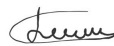
**Отчет о программном проекте**

на тему: \_\_\_\_\_ Нейросети с нуля \_\_\_\_\_

(промежуточный, этап 1)

**Выполнил:**

Студент группы БПМИ204 \_\_\_\_\_



Подпись

Гимранов Артур Маратович \_\_\_\_\_

И.О.Фамилия

03.02.2022 \_\_\_\_\_

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 \_\_\_\_\_ 2022

Оценка (по 10-ти бальной шкале) \_\_\_\_\_

Подпись

Москва 2022

# Содержание

<b>1</b>	<b>Введение</b>	<b>3</b>
<b>2</b>	<b>Важный пример</b>	<b>3</b>
<b>3</b>	<b>Функциональные требования</b>	<b>4</b>
<b>4</b>	<b>Нефункциональные требования</b>	<b>4</b>
<b>5</b>	<b>Содержательная часть</b>	<b>4</b>
5.1	Теория . . . . .	4
5.2	Net . . . . .	4
5.3	ComputeBlock . . . . .	5
5.4	Функции активации . . . . .	5
5.4.1	Sigmoid . . . . .	5
5.4.2	Relu . . . . .	5
5.4.3	Softmax . . . . .	5
5.5	Функция потерь . . . . .	6
5.6	Тестирование . . . . .	6
5.6.1	Линейные преобразования . . . . .	6
5.6.2	Нелинейные функции . . . . .	6
5.7	Mnist digits . . . . .	6

## Аннотация

В рамках этого проекта предполагается изучить теорию нейросетей и их обучения и написать проект в котором будут реализованы все необходимые компоненты для работы и обучения нейросети.

## 1 Введение

В проекте я реализую следующие компоненты: узел нейросети, который состоит из линейного отображения и нелинейной части, объекты отвечающие функциям штрафа. Будет реализован механизм вычисления градиента для узла и проталкивания градиента в узлы предыдущего слоя. На основе этого механизма будут написаны методы обучения нейросети. Таким образом передо мной стоят следующие задачи:

1. изучить теорию нейросетей и градиентного спуска
2. кратко изложить в отчете теорию
3. имплементировать необходимые классы и структуры
4. изложить в отчете архитектуру и дизайн имплементации
5. написать сопроводительную документацию.

Теперь про общую постановку задачи. У нас есть  $k$  переменных  $x = [x^{(1)}, \dots, x^{(k)}]^t$ , через которые мы хотим выразить переменную  $y$  в виде некоторой функции:

$$f(x) = y$$

Понятно, что для  $n$  векторов и образов эта задача решается очень легко, нам же хочется обучить модель по начальной выборке, чтобы ошибка на других векторах была минимальна

Как будем искать приближение? Давайте для простоты рассмотрим двухслойную нейросеть. Нейроном будем называть блок в который мы подаем вектор  $\in \mathbb{R}^n$ , а он нам возвращает вектор  $\in \mathbb{R}^m$ , и сам обладает каким-то набором параметров  $\theta$ , будем говорить что это такая функция

$$f(x, \theta) : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

Теперь разберем пример из двух блоков. Пусть у нас есть два блока с параметрами  $\theta_1, \theta_2$  и у нас выполняется такая цепочка функций

$$\mathbb{R}^n \xrightarrow{x_i} \boxed{\theta_1} \xrightarrow{w_i} \boxed{\theta_2} \xrightarrow{z_i} \mathbb{R}^m$$

$f(x, \theta_1) \qquad g(x, \theta_2)$

Тогда мы хотим подобрать такие параметры  $\theta_1, \theta_2$ , чтобы минимизировать ошибку на обучающей выборке  $x_i \in \mathbb{R}^n, y_i \in \mathbb{R}^m$ .

$$\begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix}$$

Функция ошибки  $\phi(\theta_1, \theta_2) = \sum_{i=1}^p \|g(f(x_i, \theta_1), \theta_2) - y_i\|^2 \rightarrow \min$ . Теперь мы хотим посчитать градиент по параметрам  $\theta_1, \theta_2$  и градиентным спуском минимизировать ошибку

## 2 Важный пример

Рассмотрим задачу. Допустим мы хотим найти зависимость цены квартиры от некоторого набора параметров: жилой площади, расстояния до метро, расстояния до центра. Представьте, что мы измерили все эти параметры для  $n$  квартир и получили наборы значений. Цену сложим в  $y_i$ , а параметры в вектора  $x_i = [x_i^{(1)}, x_i^{(2)}, x_i^{(3)}]^t$ . И мы хотим подобрать функцию  $f$ , чтобы  $f(x_i) = y_i$ . Конечно нет строгой зависимости подходящей любой квартире, но мы можем подобрать функцию в некотором виде, для которой отклонение в наших точках было бы наименьшим:

$$\sum_{i=0}^n |f(x_i) - y_i| \rightarrow \min$$

### 3 Функциональные требования

Наша программа будет получать на вход обучающую выборку подбирать по ней параметры и вычислять предполагаемое значение в точке.

1. **Net**. Инициализирует ресурсы, слои нейронки и блок функции ошибок. Главный класс проекта, нужен для обучения и предсказания значения в точке.
2. **ComputeBlock**. Те самые слои нейронки или наши "блоки". Содержит функцию и ее параметры. Умеет вычислять функцию в точке, считать градиент по параметрам и проталкивать его в следующий блок.
3. **LosFunction**. Функция ошибок, нужна для расчета отклонения и вычисления градиента

### 4 Нефункциональные требования

- C++20 [1]
- Google C++ Style Guide [2]
- GNU GCC compiler [3]
- ClangFormat linter [4]
- Библиотеки: Eigen [5], glm [6], cuBlas [7]
- Система поддержки версий: git [8] с github [9]

### 5 Содержательная часть

#### 5.1 Теория

Сеть

$$\mathbb{R}^n \rightarrow \boxed{\theta_1}_{f_1(x, \theta_1)} \rightarrow \dots \rightarrow \boxed{\theta_i}_{f_i(x, \theta_i)} \rightarrow \dots \rightarrow \boxed{\theta_k}_{f_k(x, \theta_k)} \rightarrow \mathcal{L} \rightarrow \mathbb{R}$$

Где  $f_i = \phi(A_i x + b_i)$ ,  $\phi(x)$  – функция активации,  $(A_i, b_i) = \theta_i$  – параметры блока,  $\mathcal{L}$  – функция потерь. Пусть  $F_\Theta(x)$ ,  $\Theta = (\theta_1, \dots, \theta_k)$  функция предсказания, то есть  $F_\Theta$  передает выход  $i$  блока на вход  $i + 1$ , пока не дойдет до последнего блока и не посчитает предсказываемый результат. Тогда функция которую мы хотим минимизировать  $\psi(\Theta) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(F_\Theta(x_i), y_i)$ .

Цель заключается в том, чтобы пройти вперед, посчитать предсказание, отдать его в функцию потерь, и посчитать градиент по параметрам обратным проходом. Нам надо уметь считать  $\frac{\partial \psi}{\partial \theta_i}$

**Короче не знаю надо ли подробно расписывать про подсчет градиентного спуска**

#### 5.2 Net

Главный класс сети. Содержит в себе слои и функцию потерь. Распределяет все данные между слоями. Выполняет обучение сети с последующим предсказанием результата.

В конструкторе мы задаем размеры слоев, функцию активации для каждого слоя, количество итераций обучения, размер батча для стохастического градиентного спуска и шаг градиентного спуска

```
1 Net(const std::vector<Index>& layers_sizes, const std::vector<std::string>& layers_types,
2     EpochType epoch, BatchSizeType batch_size, LearningRateType lr);
```

## 5.3 ComputeBlock

Слой нейронной сети. Содержат параметры  $\theta = (A, b)$  и функцию активации  $\phi$ . Нужен для вычисления функции  $f(x) = \phi(Ax + b)$ , где  $x$  приходит на вход функции `evaluate` и градиентов  $\frac{\partial f}{\partial A}, \frac{\partial f}{\partial b}, \frac{\partial f}{\partial x}$ . Также обновляет параметры по заданному шагу спуска и размеру батча

```
1 void update_parameters(LearningRateType lr, BatchSizeType batch_size) {
2     A_ -= lr * dA_ / batch_size;
3     b_ -= lr * db_ / batch_size;
4 }
```

В конструкторе получает функцию активации, размеры матрицы и инициализирует  $A$  и  $b$  случайными числами из равномерного распределения на  $[-1, 1]$

## 5.4 Функции активации

Здесь хочу описать класс от которого наследуюсь в функторах, но ты сказал, что так не стоит делать

### 5.4.1 Sigmoid

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

Популярная раньше функция активации. Имеет проблему затухания градиента

```
1 Matrix evaluate(const Matrix& x) override {
2     return (1 / (1 + exp(-x.array()))).matrix();
3 }

1 Matrix derivative(const Vector& x) override {
2     return ((exp(-x.array())) / pow(exp(-x.array()) + 1, 2)).matrix().asDiagonal();
3 }
```

### 5.4.2 Relu

$$\text{Relu}(x) = \max(x, 0)$$

```
1 double relu(double x) {
2     if (x > 0) {
3         return x;
4     }
5     return 0.01 * x;
6 }

1 double relu_derivative(double x) {
2     if (x > 0) {
3         return 1;
4     }
5     return 0.01;
6 }
```

Чтобы решить проблему умирающего ReLU, в отрицательных значения возвращаю  $0.01x$

### 5.4.3 Softmax

$$z \in \mathbb{R}^n, \sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^n e^{z_k}}$$

```
1 Matrix evaluate(const Matrix& x) override {
2     return (exp(x.array()) / exp(x.array()).sum()).matrix();
3 }
```

```

1  Matrix derivative(const Vector& x) override {
2      size_t n = x.rows();
3      Matrix ans(n, n);
4      Vector eval = evaluate(x);
5      for (size_t i = 0; i < n; ++i) {
6          for (size_t j = 0; j < n; ++j) {
7              if (i == j) {
8                  ans(i, j) = eval(i) * (1 - eval(j));
9              } else {
10                 ans(i, j) = -eval(i) * eval(j);
11             }
12         }
13     }
14     return ans;
15 }
16

```

## 5.5 Функция потерь

В качестве функции потерь была взята  $L_2$  норма

## 5.6 Тестирование

### 5.6.1 Линейные преобразования

TODO

### 5.6.2 Нелинейные функции

TODO

## 5.7 Mnist digits

Нейронная сеть была протестирована на популярном датасете из изображений рукописных цифр. Обучалась на 8500 примерах, и на выборке из 1500 тестов предсказала верно 88% изображений

## Список литературы

- [1] URL: <https://en.cppreference.com/w/cpp/20>.
- [2] URL: <https://google.github.io/styleguide/cppguide.html>.
- [3] URL: <https://gcc.gnu.org>.
- [4] URL: <https://clang.llvm.org/docs/ClangFormat.html>.
- [5] URL: [https://eigen.tuxfamily.org/index.php?title=Main\\_Page](https://eigen.tuxfamily.org/index.php?title=Main_Page).
- [6] URL: <https://glm.g-truc.net/0.9.9/>.
- [7] URL: <https://docs.nvidia.com/cuda/cublas/index.html>.
- [8] URL: <https://git-scm.com>.
- [9] URL: <https://github.com>.