

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

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК: 192.168.1.1

Отчет об исследовательском проекте

на тему: _____ Нейросеть с нуля _____

Выполнил:

Студент группы БПМИ211 _____

05.02.2023

Дата

Подпись

Д. А. Сорокин

И.О.Фамилия

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 12.06 2023

10

Оценка (по 10-ти бальной шкале)

Подпись

Москва 2023

Содержание

| | | |
|---|--|---|
| 1 | Введение | 2 |
| 2 | Описание структуры нейронной сети | 2 |
| 3 | Описание алгоритма обучения | 2 |
| 4 | Описание функциональных и нефункциональных требований к программному проекту | 4 |
| 5 | Инструменты, используемые в проекте | 4 |

1 Введение

Основная задача данного проекта это изучить базовые принципы работы нейронных сетей и создать собственную реализацию библиотеки для работы с нейронными сетями на языке C++. В данном случае под нейронной сетью подразумевается метод машинного обучения, который широко используется для построения моделей и прогнозирования. Нейронная сеть представляет слои, состоящие из вычислительных узлов, связанных между собой.

Спектр применения нейронных сетей крайне велик, но мы рассмотрим общую формулировку:

Есть векторные пространства $\mathbb{R}^n, \mathbb{R}^m$ и неизвестная функция $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$, так же есть множество примеров - пары $(x_i, y_i = \phi(x_i))$ наша цель научиться предсказывать $\phi(x_i)$ наиболее точно.

Результатом работы будет являться библиотека на языке C++, для построения и обучения своих нейронных сетей.

Прежде чем переходить к описанию программной части, стоит дать более точное описание нашего алгоритма.

2 Описание структуры нейронной сети

Для начала введем функцию потерь - это функция будет показывать насколько наше предсказание близко к искомой функции, например можем использовать среднее арифметическое отклонений на каждой паре тестовых данных:

$$\sum_{i=1}^n \frac{L(\psi(x_i), y_i)}{n}$$

Наша цель - минимизировать функцию потерь.

Нейронная сеть будет состоять из линейных слоев, устроенных следующим образом:

Это функция $\phi'(x) : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}, \phi'(x) = Ax + b$, где A и b являются параметрами. Также после каждого линейного слоя идет нелинейный:

$$\sigma(x) \mathbb{R}^n \rightarrow \mathbb{R}^n$$
$$\sigma \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} \sigma(x_1) \\ \sigma(x_2) \\ \dots \\ \sigma(x_n) \end{pmatrix}$$

Где $\sigma(x)$ некоторая нелинейная функция, подробнее про конкретные примеры σ в программной части.

Нелинейные слои необходимы, иначе комбинация слоев будет всегда являться линейной функцией.

Комбинация слоев и будет являться нейронной сетью. Осталось определить процесс обучения.

3 Описание алгоритма обучения

Воспользуемся градиентным спуском - идея в том, что нашу нейронную сеть можно представить как сложную функцию зависящую от многих параметров (A_i, b_i в линейных слоях) тогда если мы посчитаем производную

функции потерь от этих переменных и сдвинемся в сторону противоположную градиенту, мы уменьшим значение функции потерь. Повторив так некоторое количество раз, мы окажемся в локальном минимуме.

Теперь посчитаем градиент, основываясь только на одной паре x_i, y_i :

$\phi(F(x_i), y_i)$ - наша функция потерь, F - композиция всех слоев сети, θ - все параметры сети, $\phi(x_i) = \omega_i$

$$\begin{aligned}\frac{\delta}{\delta\theta}\phi(F(x_i), y_i) &= \frac{\delta\phi}{\delta\theta}(w_i, y_i) \frac{\delta F}{\delta\theta}(x_i) \\ \frac{\delta\phi}{\delta\theta}(w_i, y_i) &= u_i\end{aligned}$$

Рассмотрим последний слой нейронной сети, состоящий из композиции линейного и нелинейного:

$$\begin{aligned}\mathbb{R}^m &\xrightarrow{g} \mathbb{R}^n \xrightarrow{\sigma} \mathbb{R}^n \\ z_i &\rightarrow Az_i + b \rightarrow \sigma(Az_i + b)\end{aligned}$$

Где z_i - результат преобразования x_i предыдущими слоями.

$$d(\sigma(Az + b)) = d\sigma(dAz + Adz + db)$$

Вернемся к $\frac{\delta}{\delta\theta}\phi(F(x_i), y_i)$:

$$\frac{\delta}{\delta\theta}\phi(F(x_i), y_i) = u_i * d\sigma(dAz + Adz + db)$$

Перепишем это в таком формате:

$$u_i d(\sigma) d(A) z + u_i d(\sigma) A d(z) + u_i d(\sigma) db$$

Т.к это одномерное выражение справедливо:

$$u_i d(\sigma) d(A) z = tr(u_i d(\sigma) d(A) z) = tr(z u_i d(\sigma) d(A))$$

Тогда по правилам [матричного дифференцирования](#) $(z u_i d(\sigma))^T = \frac{d(F \circ \phi)}{dA}$

$$(u_i d(\sigma))^T = \frac{d(F \circ \phi)}{db}$$

И наконец $u'_i = u_i d(\sigma) A$ - значение u_i которое будет использовано для вычисления градиента в предыдущих слоях.

$d(\sigma)$ в свою очередь является диагональной матрицей в силу устройства σ и значения на диагонали вычисляются по формуле $\sigma'(Az_i + b)_i$.

Так мы научились последовательно вычислять градиент по слоям для одной пары (x_i, y_i) .

Заметим, что вычисления градиента по батчу пар не сложнее - достаточно вычислить градиент для k пар, затем сложить градиенты и поделить на k .

4 Описание функциональных и нефункциональных требований к программному проекту

Результатом проекта является библиотек для создания/обучения нейронных сетей, написанная на языке C++. Нейронная сеть состоит из следующих частей:

- **Линейный слой** - класс представления линейного слоя нейронной сети, обладающий следующими методами:
 - **CalculateByX(x)** - применяет слой к вектору x и возвращает результат вычисления
 - **CalculateDerivative(u)** - вычисляет dA, db и u' по посчитанному ранее u.
- **Нелинейный слой** - класс представления нелинейного слоя, на основе одной из нелинейных функций. Методы:
 - **CalculateByX(x)** - применяет слой к вектору x и возвращает результат вычисления
 - **CalculateDerivative(x)** - вычисляет производную
- **Класс представления нейронной сети** - создает нейронную сеть, с заданным количеством слоев и заданными нелинейными функциями. Методы:
 - **Train** - реализация градиентного спуска
- **Вспомогательные классы**
 - **Классы поддерживаемых нелинейных функций**
 - **Классы для функций потерь**

5 Инструменты, используемые в проекте

При выполнении проекта мы необходима [документация C++](#).

Так же будет использоваться библиотека для матричных вычислений [Eigen](#).

Список литературы