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: Нейросеть с нуля

Выполнил:
Студент группы БПМИ211 _____ Д. А. Сорокин
Подпись И.О.Фамилия

05.02.2023
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 12.06 2023 11 _____
Оценка (по 10-ти бальной шкале) Подпись

Москва 2023

Содержание

1 Введение

В современном информационном обществе нейронные сети стали одной из наиболее востребованных и перспективных областей искусственного интеллекта. Нейронные сети используются для решения разнообразных задач, включая распознавание образов, классификацию данных, прогнозирование и генерацию контента. Они являются основой многих инновационных технологий, таких как автономные автомобили, рекомендательные системы, медицинская диагностика и многое другое.

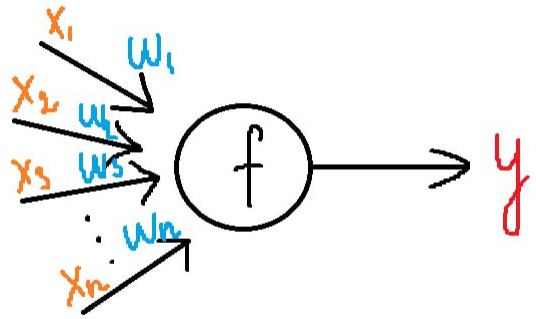
Целью данного курсового проекта является создание простой имплементации нейронной сети с использованием основных концепций и алгоритмов, лежащих в ее основе.

Результатом является написанная на языке C++ библиотека, предоставляющая функционал для создания и обучения нейронных сетей при помощи алгоритма градиентного спуска.

2 Основные идеи нейронных сетей:

Структура нейронной сети пришла в програмирование из биологии, где она описывает строение нервной ткани.

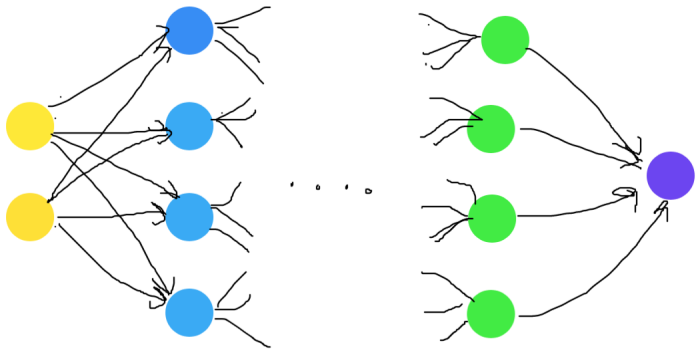
Упрощенно один нейрон можно представить следующим образом:



(1)

У него есть n входящих связей от других нейронов, при этом для каждой связи определен её вес, и один выходящий канал. Нейрон принимает данные по входящим связям и передает вычисленное значение дальше $f(x_1, ...x_n) \to y_{predicted}$.

Вместе нейроны образуют сеть, где каждый нейрон связан с другими. В нашей модели мы будем считать, что нейроны идут по слоям - каждый слой связан с предыдущим и следующим:



(2)

Процесс обучения можно описать как корректировку весов в нейронах, для получения лучшего результата.

Теперь опишем это математическим языком:

Каждый уровень сети представим в следующем виде:

$$\sigma(A \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ ... \\ x_n \end{bmatrix} + b) \tag{3}$$

- Внутренние параметры слоя - это $A \in R^{n \times m}, b \in R^n$, обозначим их за θ и нелинейная функция σ.

Рассмотрим алгоритм обучения на примере задачи предсказания стоимости дома на основе некоторых числовых параметров(таких как площадь квартиры, высота потолков etc). У нас есть данные из реального мира про некоторые квартиры с известными параметрами и ценой, мы хотим с помощью сети предсказывать цену квартиры по заданным параметрам.

Тут же встает вопрос как оценивать предсказания нашей сети, для этого введем функцию потерь, которая является дифференцируемой функцией расстояние на пространстве векторов ответов - $\mathscr{L}(y, F(x)) \to R$, где $F(x)$ - результат нашей сети. Теперь наша задача звучит как минимизировать функцию

$$\mathscr{L}'(X, Y) = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(F(x_i), y_i) \tag{4}$$

Решать эту задачу мы будем при помощи градиентного спуска - вычислим производные $\frac{\delta}{\delta \theta_i} \mathscr{L}'(X, Y)$, где θ_i - изменяемый параметр i слоя. Зная значения градиента для каждого слоя мы вычитаем его из параметров и тем самым сдвигаемся в сторону локального минимума функции потерь. Подробное описания вычисления градиента приведено ниже.

3 Основные математические выкладки:

Напомним вид нашей сети:

$$\mathbb{R}^n \underset{\sigma_1(A_1 x + b_1)}{\longrightarrow} \mathbb{R}^{n_1} \underset{\sigma_2(A_2 x + b_2)}{\longrightarrow} \mathbb{R}^{n_2} ... \underset{\sigma_k(A_k x + b_k)}{\longrightarrow} \mathbb{R}^{n_k} \underset{\mathscr{L}}{\longrightarrow} \mathbb{R} \tag{5}$$

Обозначим $\Theta = (\theta_1, \theta_2 ... \theta_k)$ соотвественно $F_\Theta(x)$ - функция всей нейронной сети. Мы хотим минимизировать

$$\mathscr{L}' = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(F_\Theta(x_i), y_i) \tag{6}$$

А точнее необходимо найти

$$\frac{\delta}{\delta \theta_j} \mathscr{L}' = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(F_\Theta(x_i), y_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{\delta \mathscr{L}(F_\Theta(x_i), y_i)}{\delta (F_\Theta(x_i))} \frac{\delta F_\Theta(x_i)}{\delta \theta_j} \tag{7}$$

Компонента $\frac{\delta \mathscr{L}(w,y)}{\delta w}$ не зависит от θ_j.

Распищем $\frac{\delta F_\Theta(x_i)}{\delta \theta_j}$, для этого введем несколько обозначений:

- $f_j(x) = \sigma_j(A_j x + b)$ - функия j слоя
- $z_j^{(i)}$ - значение для x_i, которое поступило на j слой, т.е $z_j = f_{j-1}(f_{j-2}(...f_1(x)))$

Тогда

$$\frac{\delta F_\Theta(x_i)}{\delta \theta_j} = \left(\prod_{u=j+1}^{k} \frac{\delta f_u(z_u^{(i)})}{d z_u^{(i)}} \right) \frac{\delta f_j(z_j^{(i)})}{\delta \theta_j}$$

Тогда получаем такую формулу:

$$\frac{\delta}{\delta \theta_j} \mathscr{L}' = \frac{1}{n} \sum_{i=1}^{n} \frac{\delta \mathscr{L}(z_k^{(i)}, y_i)}{\delta (z_k^{(i)})} \left(\prod_{u=j+1}^{k} \frac{\delta f_u(z_u^{(i)})}{d z_u^{(i)}} \right) \frac{\delta f_j(z_j^{(i)})}{\delta \theta_j} = \tag{8}$$

$$= \frac{1}{n} \sum_{i=1}^{n} \frac{\delta \mathscr{L}(F_\Theta(x_i), y_i)}{\delta z_{j+1}^{(i)}} \frac{\delta f_j(z_j^{(i)})}{\delta \theta_j} \tag{9}$$

Видим, что производные удобно считать в обратном порядке, т.к компонента с произведением имеет общий суффикс.

Теперь запишем явные матричные формулы производных для A_j, b_j посчитанных по одной паре (x_i, y_i), для удобства опустим индексы i:

$$\boxed{u_k} = \frac{\delta \mathscr{L}(z_k, y_i)}{\delta(z_k)}$$

Далее:

$$\frac{\delta F_\Theta(x_i)}{\delta \theta_j} = \delta(\sigma_k(A_k z_k + b_k)) = [\delta\sigma_k]([\delta A_k]z_k + A_k[\delta z_k] + [\delta b_k])$$

$$\frac{\delta}{\delta\theta_j}\mathscr{L}' = \boxed{u_k} \times \delta(\sigma_k(A_k z_k + b_k)) = \boxed{u_k} \times \boxed{\delta\sigma_k} \times (\boxed{\delta A_k} \times \boxed{z_k} + \boxed{A_k} \times \boxed{\delta z_k} + \boxed{\delta b_k})$$

Рассмотрим слагаемые по отдельности:

$$tr\left(\boxed{u_k} \times \boxed{\delta\sigma_k k} \times \boxed{\delta A_k} \times \boxed{z_k}\right) = tr\left(\boxed{z_k} \times \boxed{u_k} \times \boxed{\delta\sigma_k k} \times \boxed{\delta A_k}\right) = tr(S^T A)$$

Где $S = \left(\boxed{z_k} \times \boxed{u_k} \times \boxed{\delta\sigma_k k}\right)^T$, тогда $dA = \boxed{\delta\sigma_k^T} \times \boxed{u_k^T} \times \boxed{z_k^T}$ Для остальных уже проще:

$$db = \boxed{\delta\sigma_k^T} \times \boxed{u_k^T}$$

$$u_{k-1} = \frac{\delta \mathscr{L}(F_\Theta(x), y)}{\delta z_k} = \boxed{u_k} \times \boxed{\delta\sigma_k} \times \boxed{A}$$

$\boxed{\delta\sigma}$ - это диагональная матрица с одномерными производными на диагонали.

Для θ_{k-1} формулы будут ровно такими же, просто надо использовать вместо u_k, u_{k-1}, тем самым научились вычислять градиент для каждого слоя по одной паре x, y. Ясно, что для получения градиента по батчу (x, y) достаточно посчитать для каждой пары и затем взять среднее арифметическое для каждого слоя.

4 Функциональные требования:

Результатом проекта является библиотек для создания/обучения нейронных сетей, состоящая из следующих компонент:

1. NeuralNetwork - основной класс нейронный сети, инициализирует слои нейронной сети и функцию потерь. Есть метод обучения по выборке.

2. Layer - класс представления одного слоя нейронной сети, состоит из линейной части и функции активации.

3. LossFunction - класс интерфейс функции потерь нейронной сети. Есть имплементация квадратичной функции потерь.
4. ActivationFunction - класс интерфейс функции активации, реализованы следующие имплементации:
    a Relu
    b Sigmoid
    c Softmax

5 Нефункциональные требования:

- C++20
- Google C++ Style Guide
- CMake 3
- Eigen - библиотека для работы с матрицами
- git - система контроля версий

6 Детальное описание внешних классов:

6.1 NeuralNetwork

Список литературы