МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (национальный исследовательский университет)»

Институт естественных и точных наук

Кафедра прикладной математики и программирования

Отчет по лабораторной работе № 1

Реализация метода обратного распространения ошибки для двухслойной полностью связанной нейронной сети

по дисциплине

«Современные нейросетевые технологии»

ЮУрГУ - 01.04.02. 2025. 306/010. Р

Руководитель, преподаватель

______________/ Д.М. Кичеев /

«_____» мая 2025 г.

Автор

студент группы ИЕТН - ЕТ-122

_____________/ О.В. Ростова /

«_____» мая 2025 г.

Челябинск 2025

Цель настоящей работы состоит в том, чтобы изучить метод обратного распространения ошибки для обучения глубоких нейронных сетей на примере двухслойной полностью связанной сети (один скрытый слой).

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, мною были выполнены следующие шаги:

1. Изучена **общая схема метода обратного распространения ошибки**.

Метод обратного распространения ошибки (BackPropagation) - один из самых распространенных алгоритмов обучения нейронной сети. Он используется для эффективного обучения нейронной сети.

Мною изучены основные понятия.

Нейронная сеть содержит следующие слои:

- **Входной слой** - получает данные (например, фотографии улиц).
- **Скрытые слои** - обрабатывают информацию (нейроны в этих слоях имеют «**веса**» - это как их «настройки»).
- **Выходной слой** - выдаёт ответ (например, имеются ли на фотографиях изображения дорожных знаков).

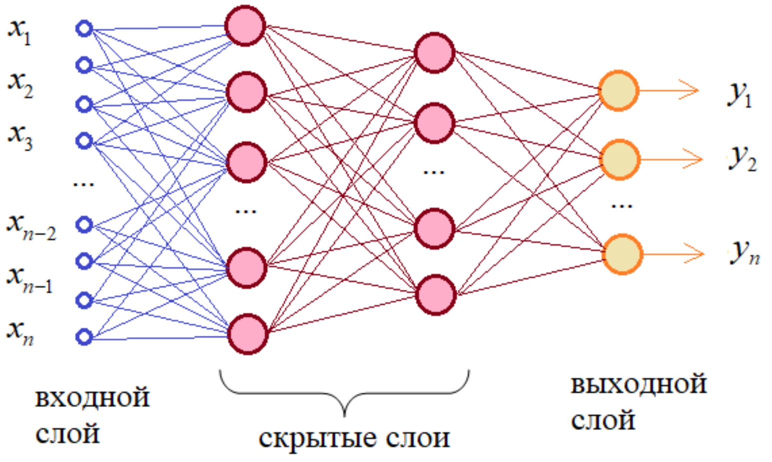

Рисунок 1 – Графическое изображение нейронной сети

Нейроны оперируют числами в диапазоне [0,1] или [-1,1]. Если числа выходят из данного диапазона, для их обработки 1 делят на это число, этот процесс называется **нормализацией**, и он очень часто используется в нейронных сетях.

Прямое распространение (Forward Pass):

Шаг 1: Данные идут через сеть от входа к выходу.

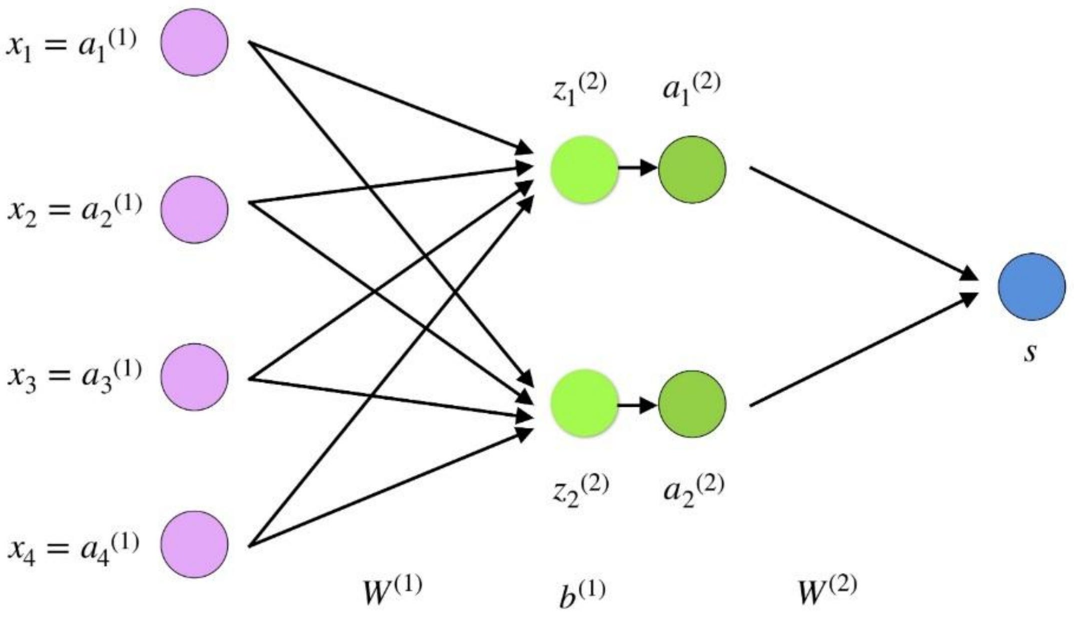

Рисунок 2 – Графическое изображение двухслойной полностью связанной нейронной сети (один скрытый слой)

На рисунке нейроны фиолетового цвета - входные данные. Они могут быть простыми скалярными величинами или более сложными – векторами или многомерными матрицами.

$x_i = a_i^{(1)}, i \in \{1, 2, 3, 4\}$.

Конечные значения в скрытых нейронах (на рисунке зеленого цвета) вычисляются с использованием z^l - взвешенных входов в слое I и a^I активаций в слое L. Для слоя 2 уравнения будут следующими:

$z^{(2)} = W^{(1)} x + b^{(1)}$

$a^{(2)} = f\ (z^{(2)})$

где $W^{(1)}$ – это веса на слое 2,

$b^{(1)}$ – смещение на этом слое.

Шаг 2: Каждый нейрон умножает входы на свои «веса», суммирует их и применяет **функцию активации** (например, softmax).

$$\text{softmax}(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}}, j = 1, \dots, k$$

где k - число выходных нейронов (классов),

z_j - взвешенная сумма на j-м выходе.

Шаг 3: На выходе получается предсказание сети.

Пример: Если сеть выдала ответ, что на фотографии нет дорожных знаков, а на самом деле они есть, это **- ошибка**.

Функция потерь (Loss Function):

Это своего рода «штраф» за ошибку. Чем больше ошибка, тем выше штраф. Пример: Если сеть уверена, что на фотографии нет дорожных знаков (90%), а правильный ответ – что они есть, штраф будет большим.

Обратное распространение ошибки (Backward Pass):

В процессе использования метода обратного распространения ошибки требуется понять, какие веса виноваты в ошибке, и скорректировать их.

Шаг 1: Вычисляем **градиенты** (производные) функции потерь по каждому весу. Градиент показывает, насколько нужно изменить вес, чтобы уменьшить ошибку.

Шаг 2: Идём от выхода к входу (поэтому «обратное» распространение).

- Для выходного слоя градиенты считаются напрямую.

- Для скрытых слоёв используем цепное правило (распространяем градиенты через предыдущие слои).

Шаг 3: Обновляем веса по формуле:

$$\omega_{new} = \omega - \eta * \partial loss/\partial w$$

где ω - текущий вес;

η - скорость обучения, т.е. шаг, с которым мы корректируем веса (например, 0,001);

а $\partial loss/\partial w$ - производная функции потерь по весу.

После каждого прохода по сети **обратное распространение** выполняет проход в обратную сторону и регулирует такие параметры модели, как **веса** и **смещения**.

Простой пример: Данные: X = 2 (вход), правильный ответ Y = 10.

Нейрон: $y = \omega * X$ (простое умножение на вес ω).

Предсказание: Если $\omega = 3 \rightarrow y = 6$. Ошибка: 10 - 6 = 4.

Функция потерь: $L = (10 - y)^2 = 16$.

Градиент: $dL/dw = 2*(10 - y)(-X) = 24*(-2) = -16$.

Обновление веса (скорость обучения = 0,01):

$\omega = 3 – 0{,}01*(-16) = 3 + 0{,}16 = 3{,}16$.

Новое предсказание: 3,16*2 = 6,32 (стало ближе к 10). Повторяем процесс.

Визуальное представление обратного распространения в нейронной сети:

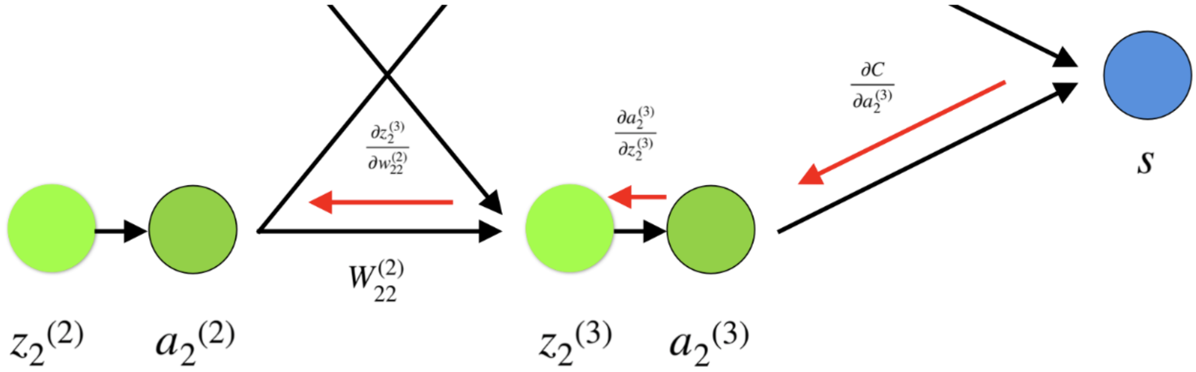

Рисунок 3 – Графическое изображение обратного распространения в нейронной сети трехслойной полностью связанной нейронной сети

На рисунке 3 изображено вычисление градиента C относительно одного веса $W_{22}^{(2)}$.

Входной слой получает данные (например, изображение), **скрытый слой** содержит **s** нейронов, а выходной слой - **k** нейронов, соответствующих числу классов. На выходном слое используется функция, например, **softmax**, а для измерения ошибки применяется **кросс-энтропия**.

Нейронная сеть функционирует не по какому-либо жестко заданному на этапе проектирования алгоритму, она **совершенствуется** в процессе анализа имеющихся данных. Этот процесс называется **обучением нейронной сети**. Математически суть процесса обучения заключается в **корректировке значений весов синапсов** (связей между имеющимися нейронами). Изначально значения весов задаются случайно, затем производится обучение, результатом которого будут новые значения синаптических весов.

Один полный проход по всей выборке называется **эпохой**. Обучение нейронной сети - это процесс, требующий многократных экспериментов, анализа результатов и творческого подхода.

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

Введем следующие обозначения:

l - количество слоев;

z - входные данные слоя;

$z^{(l)}$ - вектор «входов» (до нелинейности) l-го слоя;

W – веса слоя,

$W^{(l)}$ - матрица весов, связывающая слои (l−1) → l,

a - выходные данные предыдущего слоя

$a^{(l)}$ - вектор активаций (выходов) l-го слоя;

$a^{(l-1)}$ - выходные данные предыдущего слоя;

b - смещение слоя

$b^{(l)}$ - вектор смещений (bias) l-го слоя;

σ - выбранная функция активации (например, сигмоида, ReLU или softmax),

$C(a^{(l)}, y)$ - функция ошибки (cost), где y - «правильный» (целевой) вектор.

Выход на каждом слое l вычисляется по формуле:

$z^{(l)} = W^{(l)} * a^{(l-1)} + b^{(l)}$

Если слой – единственный, формула будет выглядеть следующим образом:

$z = W * X + b$

Входные данные в слое связаны через функцию активации:

$$a^{(l)} = \sigma (z^{(l)})$$

Функция активации softmax (по условиям задачи):

$$\text{softmax}(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{k} e^{z_i}}, \; j = 1, \dots, k$$

где k - число выходных нейронов (классов), в нашем случае их 10,

z_j - взвешенная сумма на j-м выходе.

Для задач классификации в качестве **функции ошибки** можно использовать кросс-энтропию, или функцию отрицательной логарифмической вероятности.

$$\text{Функция ошибки } L = -\sum_{i=1}^{k} Y_k \log(z_k)$$

По факту функции ошибки - сумма произведений вероятности истинного класса и логарифма вероятности предсказанного класса, отраженная со знаком «-».

Если нейросеть правильно определяет класс, то значение Z_k этого класса будет стремиться к 1, ошибка будет минимальной. И наоборот. Чем хуже предсказывает нейросеть, тем меньше значение Z_k и больше ошибка.

Производная функции ошибки L слоя l по выходным данным z:

$$\delta^{(l)} = \partial L / \partial Z^{(l)}$$

Найдем производную softmax:

Производная внутри одной и той же компоненты:

$$\frac{\hat{y}_i}{\partial z_k} = \hat{y}_i (1 - \hat{y}_i)$$

Производная между разными компонентами:

$$\frac{\partial \hat{y}_i}{\partial z_k} = - \hat{y}_i * \hat{y}_k$$

Производная кросс-энтропии:

Формула ошибки кросс-энтропии:

Функция ошибки $L = -\sum_{i=1}^{k} y_i * \log(\hat{y}_i)$

Производная ошибки до выхода до применения softmax:

$$\frac{\partial L}{\partial z_k} = \sum_{i=1}^{k} \frac{\partial L}{\partial \hat{y}_i} * \frac{\partial \hat{y}_i}{\partial z_k}$$

Частная производная функции ошибки по предсказанным вероятностям:

$$\frac{\partial L}{\partial \hat{y}_i} = -\frac{y_i}{\hat{y}_i}$$

Подставляем в формулу, получаем:

$$\frac{\partial L}{\partial z_k} = \sum_{i=1}^{k} \left(-\frac{y_i}{\hat{y}_i}\right) * \frac{\partial \hat{y}_i}{\partial z_k}$$

Подставляем в эту формулу для производной softmax.

Для разных компонент:

$$-\frac{y_k}{\hat{y}_k} * \hat{y}_k (1 - \hat{y}_k) = -y_k (1 - \hat{y}_k)$$

Для одинаковых компонент:

$$-\frac{y_i}{\hat{y}_i} * (-\hat{y}_i * \hat{y}_k) = y_i * \hat{y}_k$$

Объединим оба случая:

$$\frac{\partial L}{\partial z_k} = -y_k (1 - \hat{y}_k) + \sum_{i \neq k} y_i * \hat{y}_k$$

Так как в нашем случае y = 0, то остается:

$$\partial^{(2)} = \frac{\partial L}{\partial z^{(2)}} = \hat{y} - y$$

Расчет градиентов функции ошибки по параметрам выходного слоя:

Линейная комбинация выходного слоя:

$z^{(2)} = W^{(2)} * a^{(1)} + b^{(2)}$

Используем цепное правило дифференцирования:

$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial W^{(2)}}$$

Производная для каждого выхода нейрона на выходном слое по весу равна:

$$\frac{\partial z^{(2)}}{\partial W^{(2)}} = a^{(1)}$$

Подставляем функции:

$$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} * (a^{(1)})^T$$

$\delta^{(2)}$ – вектор ошибки на выходном слое;

$(a^{(1)})^T$ – вектор выходов скрытого слоя.

Формула для смещения:

$$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)}$$

Расчет ошибки скрытого слоя:

Производная ошибки по входу скрытого слоя:

$$\frac{\partial L}{\partial z^{(1)}} = \frac{\partial L}{\partial a^{(1)}} * \frac{\partial a^{(1)}}{\partial z^{(1)}}$$

Функция ошибки L зависит от выхода слоя через веса и входные данные:

$$\frac{\partial L}{\partial a^{(1)}} = \frac{\partial L}{\partial z^{(2)}} * \frac{\partial z^{(2)}}{\partial a^{(1)}}$$

Производная входа в скрытый слой от выхода 1 слоя:

$$\frac{\partial z^{(2)}}{\partial a^{(1)}} = W^{(2)}$$

Объединяем формулы:

$$\frac{\partial L}{\partial a^{(1)}} = (W^{(2)})^T * \delta^{(2)}$$

Переход от входных данных к выходным осуществляется через функцию активации:

$$\frac{\partial a^{(1)}}{\partial z^{(1)}} = \sigma' (z^{(1)})$$

Объединяем формулы:

$$\delta^{(1)} = \frac{\partial L}{\partial z^{(1)}} = (W^{(2)})^T * \delta^{(2)} * \sigma' (z^{(1)})$$

Расчет градиентов функции по параметрам скрытого слоя:

Вывод для выходного слоя:

Для весов:

$$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} * x^T$$

Для смещения:

$$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)}$$

Обновление параметров с использованием градиентного спуска:

Для весов скрытого слоя:

$$W^{(1)} = W^{(1)} - \eta * \frac{\partial L}{\partial W^{(1)}}$$

Для смещения скрытого слоя:

$$b^{(1)} = b^{(1)} - \eta * \frac{\partial L}{\partial b^{(1)}}$$

Для весов выходного слоя:

$$W^{(2)} = W^{(2)} - \eta * \frac{\partial L}{\partial W^{(2)}}$$

Для смещения выходного слоя:

$$b^{(2)} = b^{(2)} - \eta * \frac{\partial L}{\partial b^{(2)}}$$

Метод обратного распространения ошибки:

Рассчитаем вход скрытого слоя:

$$z^{(1)} = W^{(1)} * x + b^{(1)}$$

Используем функцию активации:

$$a^{(l)} = \sigma\, (z^{(l)})$$

Вычисляем входные данные итогового слоя:

$$z^{(2)} = W^{(2)} * a^{(1)} + b^{(2)}$$

Применяем softmax:

$$\text{softmax}(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

Вычисляем функцию ошибки:

Функция ошибки $L = -\sum_{i=1}^{k} Y_k \log(z_k)$

Обратное распространение ошибки:

Вычисляем ошибку выходного слоя:

$$\delta^{(2)} = \frac{\partial L}{\partial z^{(2)}} = \hat{y} - y$$

Применяем ошибку скрытого слоя:

$$\delta^{(1)} = \frac{\partial L}{\partial z^{(1)}} = (W^{(2)})^T * \delta^{(2)} * \sigma'(z^{(1)})$$

Вычисляем градиенты:

Для весов скрытого слоя:

$$\frac{\partial L}{\partial W^{(1)}} = \delta^{(1)} * x^T$$

Для смещения скрытого слоя:

$$\frac{\partial L}{\partial b^{(1)}} = \delta^{(1)}$$

Для весов выходного слоя:

$$\frac{\partial L}{\partial W^{(2)}} = \delta^{(2)} * (a^{(1)})^T$$

Для смещения выходного слоя:

$$\frac{\partial L}{\partial b^{(2)}} = \delta^{(2)}$$

Обновление параметров с использованием градиентного спуска:

Для весов скрытого слоя:

$$W^{(1)} = W^{(1)} - \eta * \frac{\partial L}{\partial W^{(1)}}$$

Для смещения скрытого слоя:

$$b^{(1)} = b^{(1)} - \eta * \frac{\partial L}{\partial b^{(1)}}$$

Для весов выходного слоя:

$$W^{(2)} = W^{(2)} - \eta * \frac{\partial L}{\partial W^{(2)}}$$

Для смещения выходного слоя:

$$b^{(2)} = b^{(2)} - \eta * \frac{\partial L}{\partial b^{(2)}}$$

Указанные шаги для каждого примера из обучающей выборки нужно повторять до тех пор, пока сеть не обучится.

3. Произведем проектирование и разработку программной реализации.

Реализация произведена в отдельном файле с использованием языка программирования Python в Google Colab, перенесена в GitHub.

В процессе проектирования и разработки программы реализованы описанные ранее формулы: произведена инициализация параметров, прямое распространение, обратное распределение ошибки. Произведен расчет производной функции активации, это позволяет учитывать влияние функции активации ReLU на градиенты. Произведено обновление параметров сети.

4. Проведем тестирование разработанной программной реализации.

Реализация также произведена в отдельном файле с использованием языка программирования Python в Google Colab, перенесена в GitHub.

Загружены данные базы MNIST, разделены на тестовую и тренировочную выборку. Произведена предобработка данных и их визуализация, обучение и тестирование модели с разными параметрами. Результаты представлены графически. Наилучший вариант - модели со средними параметрами, здесь оптимальна скорость обучения и точность.

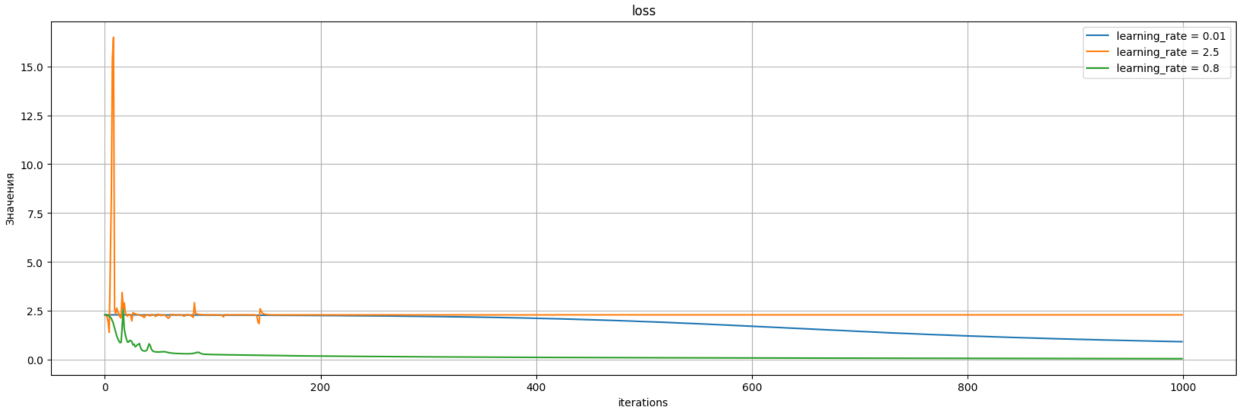

Рисунок 4 – График значений ошибки

Таким образом, в процессе выполнения лабораторной работы мною изучены изучен метода обратного распространения ошибки, произведено его пошаговое описание с выводом математических формул для сети. Разработана программная реализация метода и приложение для решения задачи классификации рукописных цифр на примере базы MNIST.

В данном отчете отражено краткое описание разработанного программного кода. Программный код выложен в личном репозитории на GitHub. В данном отчете отражены результаты классификации для тестового набора данных MNIST.