МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра: Алгебры, геометрии и дискретной математики**

Направление подготовки: «Прикладная математика и информатика»

Магистерская программа: «Компьютерные науки и приложения»

**Отчет**

**По практической работе №1**

**на тему:**

**Реализация метода обратного распространения ошибки для двухслойной полностью связанной нейронной сети**

**Выполнила:** студентка группы

381803-4М

Новожилова

Анастасия Александровна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

**Проверил:** к.т.н, доцент

Кустикова

Валентина Дмитриевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись

Нижний Новгород  
2019

Оглавление

[**Цели** 3](#_Toc24575942)

[**Задачи** 3](#_Toc24575943)

[**Задача распознавания рукописных цифр** 3](#_Toc24575944)

[**Структура сети** 3](#_Toc24575945)

[**Метод обратного распространения ошибки** 4](#_Toc24575946)

[**Описание программной реализации** 6](#_Toc24575947)

[**Результаты** 6](#_Toc24575948)

**Цели**

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).

**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.

**Задача распознавания рукописных цифр**

В процессе изучения библиотек глубокого обучения рассматривается задача распознавания рукописных цифр на изображениях. Для обучения и тестирования разработанных нейронных сетей используется широко известная база изображений MNIST. Изображения в данной базе имеют разрешение 28×28 пикселей и хранятся в формате оттенков серого. Тренировочный набор состоит из 60000 изображений, тестовый – из 10000 изображений.

**Структура сети**

Структура представляет собой двуслойную нейронную сеть прямого распространения. На входе нейронной сети имеется изображение фиксированного размера 𝑤×ℎ, содержащее рукописную цифру. Поскольку решается задача распознавания, то каждому изображению необходимо поставить в соответствие вектор достоверностей того, что изображение принадлежит одному из 10 классов {0,1,2,…,9}. Как следствие, на выходном слое нейронной сети содержится 10 нейронов. В качестве минимизируемой функции ошибки используется кросс-энтропия.

**Метод обратного распространения ошибки**

Метод обратного распространения ошибки разрабатывается, исходя из следующих предположений:

1. На входе сети имеется 𝑤 × ℎ нейронов, что соответствует разрешению изображения. В нашем случае w=h=28.

2. На выходе сети имеется 𝑘 нейронов, что соответствует количеству классов изображений. k=10 – такое количество классов изображений.

3. Скрытый слой содержит 𝑠 нейронов.

4. В качестве функции активации на первом слое используется ReLU, на втором слое - softmax.

5. В качестве функции ошибки используется кросс-энтропия.

Цель метода обратного распространения ошибки - оптимизировать весовые коэффициенты, чтобы нейронная сеть могла научиться правильно сопоставлять произвольные входные данные с выходными.

Описание алгоритма:

1. Инициализируем веса случайным образом из распределения или нулевыми значениями.
2. Прямой проход: от входного слоя, далее скрытый и выходной слой. У нас есть входные данные для скрытого слоя, считаем выходные сигналы на скрытом слое, используя соответствующую функцию активации, затем повторяем процесс с нейронами выходного слоя.
3. Теперь вычисляем значения производных функций активации на каждом слое сети.
4. Обратный проход: обновляем веса в сети так, чтобы они приводили к тому, чтобы фактический выходной сигнал был ближе к целевому выходному значению, минимизируя таким образом ошибку для каждого выходного нейрона и сети в целом, вычисляя градиент целевой функции .
5. Повторение шагов 2-4, пока не сработает критерий остановки по достигнутой точности или по числу итераций.

Представим формулы функций активации, которые будем использовать в будущей модели.

На скрытом слое используем ReLU:

Так как у нас число классов равно 10, что достаточно немало, то, чтобы выход модели отражал вероятность этих классов, и сумма вероятностей по выходам сети равнялась единице, используется softmax.

, где n – количество классов.

Функция, оценивающая качество работы всей модели, это функция потерь. В нашем случае будет cross-entropy.

.

Функция потерь является своего рода количественным “штрафом”, который можно рассматривать как меру качества прогноза модели. Это значение мы и будем использовать для обучения модели с помощью backpropagation — метода обратного распространения ошибки. Формулы, которые используют эту ошибку и “протягивают” ее сквозь все слои для обновления параметров и обучения модели.

Формула для обратного распространения ошибки через функцию потерь –

Cross-entropy по выходному слою:

Через функцию активации – softmax (так как функция softmax для i-того выхода зависит не только от своего , но и от всех других  Поэтому и формула для backpropagation “распадается” на две: частная производная по и ):

Вывод формулы для обновления весов:

**Описание программной реализации**

Данные MNIST поступают на вход программе с помощью mlxtend.data, импортирующей loadlocal\_mnist. Прежде чем обучать сеть, необходимо предобработать данные. Для этого преобразуем их в вектор и нормализуем. Инициализируем сеть:

*net = Network(num\_nodes\_in\_layers = [784, 300, 10], batch\_size = 16, num\_epochs = 20,*

*learning\_rate = 0.1)*

Параметры обучения следующие:

*num\_nodes\_in\_layers = [784, 300, 10]* – 784 – размерность входного слоя (28 \* 28), 300 – размерность скрытого слоя, 10 – размерность выходного слоя (10 цифр).

*batch\_size* – размер пакета

*num\_epochs –* количество эпох (сколько итераций проводится для обучения)

*learning\_rate –* скорость обучения

Используем Xavier initialization для инициализации начальных весов, перемешиваем выборку. Тренировочные данные делятся на batches, и для каждого такого пакета выполняется алгоритм backpropagation. Обучив модель, мы можем проверить ее на тестовых данных, также подсчитать потери и точность.

**Результаты**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Experiment №** | **Number hidden nodes** | **Batch size** | **Number**  **epochs** | **Learning rate** | **Training accuracy (%)** | **Training loss** | **Test accuracy (%)** | **Test loss** | **Time (s)** |
| 1 | 300 | 16 | 20 | 0.1 | 0.983 | 0.06 | 0.9753 | 0.08 | 517.28 |
| 2 | 30 | 16 | 20 | 0.1 | 0.969 | 0.1 | 0.9636 | 0.12 | 41.52 |
| 3 | 64 | 32 | 20 | 0.1 | 0.963 | 0.12 | 0.9607 | 0.13 | 43.81 |
| 4 | 128 | 32 | 20 | 0.1 | 0.966 | 0.11 | 0.9625 | 0.12 | 85.16 |
| 5 | 300 | 64 | 20 | 0.1 | 0.971 | 0.1 | 0.9671 | 0.11 | 301.35 |