

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 математического обеспечения и суперкомпьютерных технологий**

Направление подготовки «Прикладная математика и информатика»

Магистерская программа «Системное программирование»

**Отчет по лабораторной работе**

**«Реализация метода обратного распространения ошибки для двуслойной  
полностью связанной нейронной сети»**

***Выполнил:***

студент гр. 381603м4

Волокитин В. Д.

Нижегород

2017

## Оглавление

Постановка задачи.....	3
Описание метода обратного распространения ошибки. Вывод математических формул .....	4
Общая схема метода обратного распространения ошибки .....	7
Программная реализация.....	8
Руководство пользователя.....	9
Результаты экспериментов.....	10

## Постановка задачи

### Цель работы

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

### Постановка задачи

Выполнение лабораторной 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.

Метод обратного распространения ошибки разрабатывается, исходя из следующих предположений:

1. На входе сети имеется  $w \times h$  нейронов, что соответствует разрешению изображения.
2. На выходе сети имеется  $k$  нейронов, что соответствует количеству классов изображений.
3. Скрытый слой содержит  $s$  нейронов.
4. В качестве функции активации на втором слое используется функция softmax.
5. В качестве функции ошибки используется кросс-энтропия.

## Описание метода обратного распространения ошибки. Вывод математических формул

### Математическое объяснение метода. Постановка задачи оптимизации

Введём следующие обозначения:

- $N$  – количество входных нейронов;
- $M$  – количество выходных нейронов;
- $K$  – количество нейронов на скрытом слое;
- $L$  – количество обучающих примеров.

В качестве функции ошибки рассматривается кросс-энтропия:

$$E(w) = -\frac{1}{L} \sum_{k=1}^L \sum_{j=1}^M y_j^k \ln(u_j^k), y_j^k = 1 \leftrightarrow x^k \in j - \text{му классу},$$

где  $y^k = (y_j^k)_{j=\overline{1,M}} \in Y$  – множество обучающих примеров, а  $u^k = (u_j^k)_{j=\overline{1,M}}$  – выход нейронной сети, полученный для входного примера  $x^k = (x_i^k)_{i=\overline{1,N}} \in X$ .

Будем рассматривать последовательный (стохастический) режим обучения. В этом режиме корректировка весов выполняется после предъявления каждого примера обучающей выборки.

Возьмём конкретный обучающий пример  $x = (x_1, x_2, \dots, x_N)$ ,  $y = (y_1, y_2, \dots, y_M)$ ,  $u = (u_1, u_2, \dots, u_M)$ .

Тогда имеем

$$E(w) = -\sum_{j=1}^M y_j \ln(u_j)$$

Обозначим  $w_{is}^{(1)}$  – веса синаптических связей от входных нейронов к нейронам скрытого слоя,  $w_{sj}^{(2)}$  – от нейронов скрытого слоя к выходным нейронам.

Выходной сигнал нейрона скрытого слоя вычисляется как:

$$v_s = \varphi(f_s), \text{ где } \varphi - \text{функция активации на скрытом слое},$$

$$f_s = \sum_{i=1}^N w_{is}^{(1)} x_i, \quad s = \overline{0, K} - \text{взвешенная сумма входных сигналов}.$$

Сигнал выходного нейрона определяется как:

$$u_j = h(g_j), \text{ где } h - \text{функция активации на выходном слое},$$

$$g_j = \sum_{s=1}^K w_{sj}^{(2)} v_s, \quad j = \overline{1, M} - \text{взвешенная сумма сигналов со скрытого слоя}.$$

В качестве функции активации на последнем слое рассматривается *softmax*:

$$u_j = \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}$$

Таким образом,

$$E(w) = -\sum_{j=1}^M y_j \ln\left(\frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}\right) = -\sum_{j=1}^M y_j (g_j - \ln \sum_{m=1}^M e^{g_j}),$$

$$g_j = \sum_{s=0}^K w_{sj}^{(2)} \varphi\left(\sum_{i=0}^N w_{is}^{(1)} x_i\right).$$

Задача обучения нейронной сети сводится к задаче оптимизации функции ошибки по всем синаптическим весам  $E(w) \rightarrow \min_w$ .

### Обратное распространение ошибки

Метод обратного распространения ошибки определяет стратегию выбора параметров сети  $w$  с использованием градиентных методов оптимизации.

Производная целевой функции по параметрам последнего слоя  $w_{sj}^{(2)}$  вычисляется по следующей формуле:

$$\frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial w_{sj}^{(2)}},$$

$$\frac{\partial g_j}{\partial w_{sj}^{(2)}} = v_s,$$

В задачах классификации  $\sum_{j=1}^M y_j = 1$ , поэтому рассмотрим:

$$\delta_j^{(2)} = \frac{\partial E}{\partial g_j} = -\frac{\partial}{\partial g_j} \left[ \sum_{j=1}^M y_j (g_j - \ln \sum_{m=1}^M e^{g_j}) \right] = -(-y_1 \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - \dots - y_{j-1} \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} + y_j \left( 1 - \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} \right) - y_{j+1} \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - \dots - y_M \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}}) =$$

$$(\sum_{j=1}^M y_j) \cdot \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - y_j = \frac{e^{g_j}}{\sum_{m=1}^M e^{g_m}} - y_j = u_j - y_j.$$

В итоге получаем  $\frac{\partial E(w)}{\partial w_{sj}^{(2)}} = (u_j - y_j) \cdot v_s = \delta_j^{(2)} \cdot v_s$ .

Производная целевой функции по параметрам скрытого слоя  $w_{is}^{(1)}$  вычисляется по формуле:

$$\frac{\partial E(w)}{\partial w_{is}^{(1)}} = \frac{\partial E}{\partial f_s} \frac{\partial f_s}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i.$$

$$\frac{\partial E}{\partial f_s} = \sum_{j=1}^M \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial v_s} \frac{\partial \varphi}{\partial f_s} = \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \frac{\partial E}{\partial g_j} \frac{\partial g_j}{\partial v_s} = \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \delta_j^{(2)} w_{sj}^2$$

В итоге имеем  $\frac{\partial E(w)}{\partial w_{is}^{(1)}} = \frac{\partial \varphi}{\partial f_s} [\sum_{j=1}^M \delta_j^{(2)} w_{sj}^2] \cdot x_i$ .

Если функция активации на скрытом слое сигмоидальная  $\varphi(f_s) = \text{sigmoid}(f_s)$ , то

$$\frac{\partial \varphi}{\partial f_s} = (\varphi - \varphi^2).$$

Градиент имеет следующую структуру:  $\frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \delta_j^{(2)} \cdot v_s$ ,  $\frac{\partial E(w)}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i$ .

Согласно градиентным методам на каждом шаге  $r + 1$  обучения сети производится коррекция весов по следующим формулам:

$$w_{is}^{(1)(r+1)} = w_{is}^{(1)(r)} - \eta \frac{\partial E(w)}{\partial w_{is}^{(1)}},$$

$$w_{sj}^{(2)(r+1)} = w_{sj}^{(2)(r)} - \eta \frac{\partial E(w)}{\partial w_{sj}^{(2)}}.$$

где  $\eta$  – скорость обучения.

## Общая схема метода обратного распространения ошибки

1. Инициализировать веса  $w$  случайными значениями.
2. *for*  $eras = \overline{1, number\_eras}$
3. *for*  $i = \overline{0, w * h}$
4. Прямой проход нейронной сети в направлении передачи информации от входного сигнала к скрытому и к выходному слою сети.

### Прямой проход:

Подать на вход  $x_i$ . Вычислить значения выходных сигналов нейронов скрытого слоя  $v_s, s = \overline{0, K}, K$  – количество нейронов на скрытом слое и значение производной функции активации на скрытом слое  $\frac{\partial \varphi}{\partial f_s}$ . Вычислить выходные сигналы нейронов последнего слоя  $u_j, j = \overline{1, M}, M$  – количество классов изображений.

$$x_i \rightarrow v_s, \frac{\partial \varphi}{\partial f_s} \rightarrow u_j$$

### Обратный проход:

Вычисляются значения градиентов целевой функции.

*for*  $j = \overline{1, M}$

$$\delta_j^{(2)} = u_j - y_j, \frac{\partial E(w)}{\partial w_{sj}^{(2)}} = \delta_j^{(2)} \cdot v_s$$

*for*  $s = \overline{0, K}$

$$\delta_s^{(1)} = - \frac{\partial \varphi}{\partial f_s} \sum_{j=1}^M \delta_j^{(2)} w_{sj}^2, \frac{\partial E(w)}{\partial w_{is}^{(1)}} = \delta_s^{(1)} x_i.$$

*for* по дугам

$$w_{is}^{(1)} = w_{is}^{(1)} - \eta \frac{\partial E(w)}{\partial w_{is}^{(1)}}$$

$$w_{sj}^{(2)} = w_{sj}^{(2)} - \eta \frac{\partial E(w)}{\partial w_{sj}^{(2)}}$$

5. Повторение этапов 3 - 5 до момента выполнения критериев остановки. В качестве критериев остановки используется число итераций метода (количество проходов *number\_epochs*).

## Программная реализация

Программная реализация метода обратного распространения ошибки была разработана на языке C(/C++) в соответствии с приведённым выше алгоритмом и выведенными формулами. Запуск программы производился в Microsoft Visual Studio 2012.

В проекте *MNIST* имеются следующие файлы:

- *readHeader.h*, *readMNIST.cpp* – файлы, содержащие методы для чтения набора данных MNIST.
- *structures.h* – заголовочный файл, содержащий объявление всех структур необходимых в программе.
- *configuration.h*, *configuration.cpp* – файлы, содержащие функции для чтения конфигурационного файла config.txt.
- *neuralNetwork.h*, *neuralNetwork.cpp* – файлы, содержащий все функции для нейронной сети с возможностью обучения методом обратного распространения ошибки.
- *main.cpp* – тестовое приложение для классификации рукописных цифр на базе данных MNIST.



## Руководство пользователя

Для запуска программы необходимо задать все параметры в файле config.txt для корректной работы нейронной сети. Файл должен находиться в папке рядом с бинарным исполняемым файлом и последовательность параметров внутри файла фиксировано:

1. train\_data\_path – задает путь, абсолютный или относительный для файла с тренировочным набором изображений (например, = train-images.idx3-ubyte);
2. train\_label\_path – задает путь, абсолютный или относительный для файла с тренировочным набором подписей (например, = train-labels.idx1-ubyte);
3. test\_data\_path – задает путь, абсолютный или относительный для файла с тестовым набором изображений (например, = t10k-images.idx3-ubyte);
4. test\_label\_path – задает путь, абсолютный или относительный для файла с тестовым набором подписей (например, = t10k-labels.idx1-ubyte);
5. number\_hidden\_neurons – задает количество нейронов на скрытом слое сети (например, = 800);
6. learn\_rate – задает множитель скорости обучения (например, = 0.01);
7. number\_eras – задает количество эпох, которые будут выполнены для обучения (например, = 20).

## Результаты экспериментов

Было разработано приложение, позволяющее обучать и тестировать двухслойную нейронную сеть с использованием набора данных MNIST.

В таблице 1 собраны результаты полученной ошибки.

Таблица 1. Результаты экспериментов

Число нейронов скрытого слоя	Количество эпох	Ошибка на тренировочном наборе	Ошибка на тестовом наборе	Ошибка на сайте <a href="http://yann.lecun.com/exdb/mnist/">http://yann.lecun.com/exdb/mnist/</a> (возможны другие параметры нейронной сети)
100	20	0.0074	0.0267	-
300	20	0.006	0.0226	0.016
300	15	0.009	0.0225	0.016
800	20	0.0057	0.0219	0.011