Нижегородский государственный университет им. Н. И. Лобачевского

Институт информационных технологий, математики и механики

Направление подготовки Прикладная математика и информатика

Магистерская программа Вычислительные методы и суперкомпьютерные технологии

Образовательный курс «Методы глубокого обучения для решения задач компьютерного зрения»

**Отчёт**

по лабораторной работе № 1

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

***Выполнил:***

студент гр. 381603м4

Розанов А.

Нижний Новгород

2017

Содержание

[Постановка задачи 3](#_Toc501716583)

[Метод обратного распространения ошибки 4](#_Toc501716584)

[Обозначения 4](#_Toc501716585)

[Нейронная сеть 4](#_Toc501716586)

[Описание метода 4](#_Toc501716587)

[Программная реализация 7](#_Toc501716588)

[Класс Neuron 7](#_Toc501716589)

[Класс Layer 7](#_Toc501716590)

[Класс HiddenLayer 7](#_Toc501716591)

[Класс OutputLayer 8](#_Toc501716592)

[Класс Network 8](#_Toc501716593)

# Постановка задачи

Основной целью данной лабораторной работы является изучение метода обратного распространения ошибки для обучения глубоких нейронных сетей на примере двуслойной полностью связанной сети , ориентированной на решение задачи классификации рукописных цифр из набора данных MNIST.

В ходе выполнения данной лабораторной работы предполагается выполнение следующих подзадач:

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. Разработка программной реализации.
4. Тестирование разработанной программной реализации.

# Метод обратного распространения ошибки

## Обозначения

– вход на -й нейрон -го слоя.

– сдвиг на -м нейроне -го слоя.

– вес от -го нейрона -го слоя к -му нейрону -го слоя.

– количество нейронов на -м слое.

– значение линейной комбинации вектора весов и вектора входных значений -го нейрона -го слоя.

– выход -го нейрона -го слоя, где – функция активации.

- ожидаемый выход -го нейрона последнего слоя.

## Нейронная сеть

Будем рассматривать двухслойную полносвязную нейронную сеть с функцией активации softmax на последнем слое и с функцией ошибки кросс-энтропия.

При функции активации softmax получаем следующие выходы:

Функция ошибки кросс-энтропия имеет следующий вид:

## Описание метода

В ходе метода обратного распространения ошибки выполняется уточнение весов нейронной сети с целью уменьшения ошибки. В начальный момент веса инициализируются случайным образом. Далее для каждого примера обучающей выборки выполняется следующая процедура:

1. Прямой проход нейронной сети, на котором вычисляются выходы каждого слоя.
2. Вычисление значений целевой функции и градиента этой функции.
3. Обратный проход нейронной сети, при котором корректируются веса.
4. Повторение этапов 1-3 до момента выполнения критериев остановки (количество проходов или достигнутая точность).

Корректировка весов происходит по следующей формуле:

где , – скорость обучения, – направление в многомерном пространстве параметров нейронной сети. Направление движения совпадает с направлением антиградиента:

В соответствии с выбранной нейронной сети, целевая функция записывается следующим образом:

Найдем производную целевой функции по параметрам -го слоя:

Рассмотрим производные по последнему слою:

Так как (сумма значений выходного вектора), то

Рассмотрим производные по скрытому слою:

# Программная реализация

## Класс Neuron

Программное представление исскуственного нейрона.

Поля:

1. int numPrevNeurons – количество нейронов на предыдущем слое.
2. vector<double> weights – веса на связях от нейронов предыдущего слоя к текущему.
3. vector<double> inputs – входы от нейронов предыдущего слоя.
4. double bias – смещение для текущего нейрона.

Методы:

1. double getSum()– сумматор.
2. void updateBias(double gradient, double learnRate) – обновление смещения.
3. void updateWeights(double gradient, double learnRate) – обновление весов.

## Класс Layer

Абстрактный класс для представления слоя в нейроной сети.

Поля:

1. int numNeurons – количество нейронов на текущем слое.
2. int numPrevNeurons – количество нейронов на предыдущем слое.
3. vector<Neuron> neurons – вектор нейронов принадлежащих слою.
4. vector<double> outputs – выход слоя. Состоит из выходов каждого нейрона.

Методы:

1. virtual double activateNeuron(vector<double> &sums, double sum) – виртуальный метод активации нейронов в слое.
2. vector<double> computeOutputs() – вычисление выходов каждого нейрона.
3. void updateWeights(vector<double> &gradient, double learnRate) – обновление весов у каждого нейрона.
4. void updateBiases(vector<double> &gradient, double learnRate) – обновление смещений у каждого нейрона.
5. vector<double> computeWeightedSumErrors(vector<double> &gradient) – рассчитывает взвешенную сумму ошибок для предыдущего слоя.

## Класс HiddenLayer

Наследник класса Layer, предназначен для представления скрытого слоя. Реализует методы:

1. vector<double> computeGradient(vector<double> &layerOutputs, vector<double> &nextLayerWeightedSumErrors)
2. virtual double activateNeuron(vector<double> &sums, double sum)

## Класс OutputLayer

Наследник класса Layer, предназначен для представления последнего выходного слоя. Реализует методы:

1. virtual double activateNeuron(vector<double> &sums, double sum)
2. vector<double> computeGradient(vector<double> &outputs, vector<double> &labels)

## Класс Network

Представление исскуственной нейронной сети.

Поля:

1. HiddenLayer \*hiddenLayer – скрытый слой.
2. OutputLayer \*outputLayer – выходной слой.

Методы:

1. vector<double> test(vector<double> &inputs) – тестирование сети. Возвращает предсказанное значение.
2. double train(vector<vector<double>> trainInputs, vector<vector<double>> trainLabels, int maxEpoches, double minError, double learnRate) – обучение сети.
3. double MeanCrossEntropyError(vector<vector<double>> trainInputs, vector<vector<double>> trainLabels) – вычисление средней ошибки.
4. vector<double> forwardPropagation() – прямой проход.
5. void backwardPropagation(vector<double> &hiddenOutputs, vector<double> &outputs, vector<double> &expectredValues, double learnRate) – обратный проход.