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

Институт информационных технологий, математики и механики

Кафедра математического обеспечения и суперкомпьютерных технологий

Отчёт по учебной практике

Разработка программной реализации метода обратного распространения ошибки

Выполнил:

студент ИИТММ, гр. 381406-1

Доронин М.А.

Проверил:

ст.преп. каф. МОСТ, ИИТММ

Кустикова В.Д.

Нижний Новгород

2017

Содержание

[Введение 3](#_Toc475945152)

[Постановка задачи 4](#_Toc475945153)

[Обзор выбранного метода 5](#_Toc475945154)

[Нейронные сети 5](#_Toc475945155)

[Описание разработанного метода 7](#_Toc475945156)

[Метод градиентного спуска 7](#_Toc475945157)

[Обратное распространение ошибки 7](#_Toc475945158)

[Достоинства и недостатки метода 8](#_Toc475945159)

[Описание программной реализации 10](#_Toc475945160)

[Используемые инструменты 10](#_Toc475945161)

[Общая структура проекта 10](#_Toc475945162)

[Описание структуры программы 10](#_Toc475945163)

[Описание реализованных классов 11](#_Toc475945164)

[Результаты экспериментов 12](#_Toc475945165)

[Данные 12](#_Toc475945166)

[Показатели качества 12](#_Toc475945167)

[Результаты 12](#_Toc475945168)

[Заключение 14](#_Toc475945169)

[Литература 15](#_Toc475945170)

[Ссылки 15](#_Toc475945171)

# Введение

Машинное обучение — обширный подраздел искусственного интеллекта, изучающий методы построения алгоритмов, способных обучаться. Машинное обучение находится на стыке математической статистики, методов оптимизации и классических математических дисциплин, но имеет также и собственную специфику, связанную с проблемами вычислительной эффективности и переобучения. Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации и интеллектуальным анализом данных.

Машинное обучение — не только математическая, но и практическая, инженерная дисциплина. Несмотря на то, что эта дисциплина весьма наукоемка, чистая теория, как правило, не приводит сразу к методам и алгоритмам, применимым на практике. Чтобы заставить их хорошо работать, приходится изобретать дополнительные эвристики, компенсирующие несоответствие сделанных в теории предположений условиям реальных задач. Практически ни одно исследование в машинном обучении не обходится без эксперимента на модельных или реальных данных, подтверждающего практическую работоспособность метода.

Алгоритмы машинного обучения создавались как замена классическим алгоритмам, написание которых может быть затруднительно в силу объема данных, объема признаковых описаний и прочих причин. Или же написание четкого формализованного алгоритма, в принципе, невозможно. Таким образом можно охарактеризовать машинное обучение как область науки, описывающую способы обучения компьютера решению какой-либо задачи без явного написания самого алгоритма.

В данной работе исследуется один из методов машинного обучения – обратное распространение ошибки в нейронных сетях. Используя этот метод, решается одна из классических задач машинного обучения. Полученные результаты обучения сравниваются с существующими программными реализациями.

# Постановка задачи

В ходе учебной практики требуется:

1. Изучить классические математические методы обучения по прецедентам.
2. Изучить метод обратного распространения ошибки в нейронных сетях.
3. Создать программную реализацию метода.
4. Выбрать задачу, необходимую решить при разработке методов.
5. Провести вычислительные эксперименты, проанализировать результаты.
6. Сравнить результаты с имеющимися реализациями в сторонних библиотеках.

Программу необходимо реализовать на языке С++, и использовать систему контроля версий Git. Для генерации проекта требуется написать CMake-скрипты.

Программа должна представлять собой статическую библиотеку с реализованным методом решения задачи, а также консольные приложения для применения алгоритма на некоторой обучающей выборке и проведения тестирования обученной модели.

# Обзор выбранного метода

Задача машинного обучения представляет из себя построение алгоритма поиска общих зависимостей, закономерностей и взаимосвязей, присущих как конкретному набору ограниченных данных, так и тем объектам, которые еще не наблюдались. Данные представляют собой конечное множество прецедентов. Данные о прецеденте называют также его описанием. Совокупность всех имеющихся описаний прецедентов называется обучающей выборкой. Таким образом, задача сводится к восстановлению зависимостей по эмпирическим данным обучающей выборки.

## Нейронные сети

Нейронные сети основаны на принципе коннективизма — в них соединяется большое количество относительно простых элементов, а обучение сводится к построению оптимальной структуры связей и настройке параметров связей.

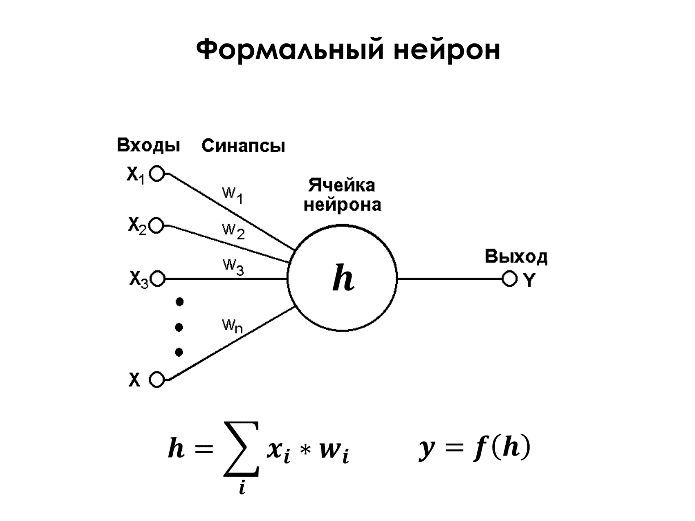
Несмотря на то, что в данный момент интерес к нейросетям достаточно высок, это один из старейших алгоритмов машинного обучения. Первый формальный нейрон, ячейка нейронной сети была предложена в 1943 году Уорреном Маккалоком и Уолтером Питтсом. Уже в 1958 году Фрэнк Розенблатт предложил первую самую простую нейронную сеть, которая уже могла разделять, например, объекты в двухмерном пространстве. Значительный интерес к нейронным сетям проявлялся в 1950–1960-е годы, когда были получены первые впечатляющие результаты. Затем нейронные сети уступили место другим алгоритмам машинного обучения, которые оказались более эффективными на момент своей разработки. Сейчас, в последние 5-7 лет оказалось, что во многих задачах, связанных с анализом естественной информации (язык, речь, изображения, видеопотоки), нейронные сети показывают лучшие результаты, по сравнению с другими алгоритмами.

На падения и взлеты интереса к нейросетям повлияли 2 причины:

1. Нейронные сети очень критичны к объему данных. При небольших размерностях обучающих выборок, сети плохо обучаются, а значит и плохо работают. Однако, в последние 15 лет данные стали легкодоступны и их объемы растут экспоненциально.
2. Нейронные сети очень требовательны к вычислительным ресурсам. Это один из самых тяжеловесных алгоритмов машинного обучения. Необходимы достаточно мощные вычислительные узлы для обучения сетей. В последние десятилетия мощности современных вычислительных систем выросли на несколько порядков, что дает возможность эффективно использовать нейронные сети.

**Формальный нейрон**

Формальный нейрон (рис. 1) представляет собой простой элемент, имеющий ограниченное число входов, к каждому из которых присвоен некоторый вес (синапсы). Работа нейрона состоит во взвешенной суммации своих входов. На входе могут быть, например, числовые значения пикселов изображения. Каждый нейрон обладает функцией активации — нелинейной функцией, ответственной за вычисление сигнала на выходе узла, аргументом которой является вычисленная взвешенная сумма. Также существует понятие смещения, представляющего из себя узел, на выходе которого всегда появляется единица.



1. Формальный нейрон [3]

**Нейронная сеть**

Нейронную сеть можно представлять в виде взвешенного направленного графа, узлы (нейроны) которого расположены слоями. При этом выходы нейронов одного слоя являются входами другого.

Как и в большинстве алгоритмов машинного обучения, цель состоит в обучении нейронной сети таким образом, чтобы достичь баланса между способностью давать верный отклик на входные данные, которые входили в обучающую выборку, и способностью выдавать правильные результаты в ответ на входные данные, схожие, но не идентичные тем, что входили в обучение (принцип обобщения).

# Описание метода обратного распространения ошибки

Для решения поставленной задачи был выбран алгоритм с использованием нейронных сетей, обучающихся методом обратного распространения ошибки (backpropagation).

## Метод градиентного спуска

Как и в любом методе обучения с учителем, на вход алгоритму предоставляется обучающая выборка , Требуется найти вектор весов , при котором минимизируется функционал ошибки . Для минимизации применим метод градиентного спуска: некоторым образом, инициализируются начальные веса, а далее итерационно веса изменяются в направлении наиболее быстрого убывания функции , то есть в направлении, противоположному направлению градиенту

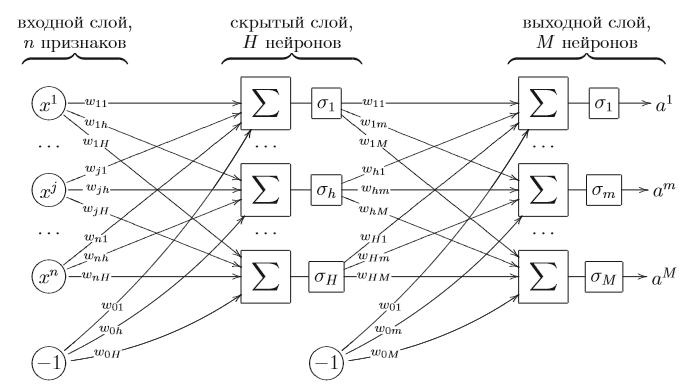
*,* где – некоторый коэффициент, регулирующий величину шага градиентного спуска (темп обучения, learning rate).

## Обратное распространение ошибки

Алгоритм обратного распространения ошибки базируется на методе градиентного спуска. В него заключены 3 этапа:

1. Подача на вход сети вектора данных, распространение сигналов сети в направлении выходов с применением взвешенного суммирования и функции активации.
2. Вычисление ошибки, то есть некоторого отклонения отклика сети и эталонного ответа.
3. Распространение ошибки от выходного слоя в направлении входного и корректировка весов связей.

Рассмотрим многослойную сеть, в которой каждый нейрон связан со всеми нейронами предыдущего слоя. Такая сеть называется полносвязной (рис. 2).



1. Нейронная сеть [4]

Пусть на входной слой подаются *n* признаков элемента выборки. Каждый входной нейрон связан со всеми нейронами скрытого слоя, а веса связей равны . Скрытый слой состоит из *H* нейронов, у каждого из которых имеется нелинейная функция активации , выходами и весам синоптических связей с выходными нейронами. В случае, если сеть двухслойная, то являются *j*-м признаком элемента выборки, и . Аналогично, выходной слой, состоящий из *M* нейронов, имеет функции активации и выходы . Для конкретного элемента обучающей выборки отклик сети вычисляется как суперпозиция:

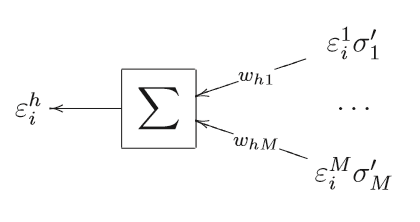
где

Вообще говоря, количество скрытых слоев может быть произвольным. Вектор ( является откликом сети, то есть ее ответом, который далее следует сравнить с эталонным и вычислить ошибку. Для этого используем функционал среднеквадратичной ошибки:

Величиной ошибки на объекте для выходного слоя тогда будут являться частные производные *Q* по :

Для вычисления ошибки на скрытом слое необходимо так же вычислить частные производные по выходам скрытого слоя:

Заметим, что ошибка на скрытых слоях вычисляется по значению ошибки на выходном слое . А значит, запустив распространение ошибки от выходного слоя к скрытым в направлении входного, получим величину ошибки на каждом нейроне. В этом и заключается принцип работы алгоритма обратного распространения ошибки.



1. Схема строения нейрона [5]

Имея частные производные и можно найти градиент по весам , то есть, по сути, величину градиентного шага.

Очевидно, что функции активации должна быть непрерывно-дифференцируема, к тому же удобнее использовать, например, сигмоидную функцию, которая имеет эффективную формулу для вычислений своей производной:

## Достоинства и недостатки метода

Алгоритм обратного распространения обладает рядом достоинств.

* Достаточно высокая эффективность метода. В случае двуслойной сети весь алгоритм, включая прямое распространение сигналов, обратное распространение ошибки и вычисление ошибок, имеет сложность порядка
* Метод приспособлен для online-обучения, то есть веса обновляются по мере поступления элементов обучающей выборки.
* Возможны различные стратегии обучения для различных объемов обучающих выборок. Так, например, для достаточно большой выборки допустимо не сохранять обучающие объекты. В случае малых выборок, можно проводить многоэпохальное обучение, то есть предъявить объекты повторно.
* Метод back-propagation предоставляет хороший ресурс для распараллеливания, так как через каждый нейрон проходит информация только о связных с ним нейронах.
* Высокая степень общности. Алгоритм можно применять для произвольного числа слоев, для разных функций активаций и разных функционалов потерь. Возможны применения различных функций активаций на разных слоях или нейронах. Кроме того, этот алгоритм хорошо сочетается с другими градиентными методами оптимизации: методом наискорейшего спуска, методом сопряженных градиентов и другими.

Недостатки алгоритма состоят в следующем:

* Алгоритм может сходиться к локальному минимуму, сходиться медленно или вовсе не сходиться. Функционал *Q* может быть многоэкстремален, и процесс градиентного спуска может остановиться в одном из локальных минимумов.
* При большом количестве признаков или при малой длине выборки может возникнуть ситуация переобучения.

# Описание программной реализации

Исходный код программы располагается в репозитории на GitHub [4].

## Используемые инструменты

В ходе работы использовались следующие инструменты:

* Система контроля версий Git.
* Среда разработки Microsoft Visual Studio (2016).
* CMake для автоматической сборки проекта.
* Jupyter – веб-оболочка для программирования на языке Python.

## Общая структура проекта

Структура проекта:

* data – директория для размещения тренировочных и тестовых данных.
* include – директория для размещения заголовочных файлов.
* sample – директория для размещения тестового приложения.
* sln – директория с файлами решений и проектов для Visual Studio 2016.
* src – директория для размещения исходных кодов (cpp-файлы).
* Служебные файлы
  + .gitignore — перечень расширений файлов, игнорируемых Git при добавлении файлов в репозиторий.

## Описание структуры программы

Программа состоит из 3 проектов:

* NeuralNetwork – статическая библиотека, в которой реализованы классы:
  + Neuron – базовый класс, описывающий сущность нейрона.
  + OutputNeuron – класс-наследник класса Neuron, описывающий нейрон выходного слоя.
  + HiddenNeuron – класс-наследник класса Neuron, описывающий нейрон скрытого слоя.
  + Function – абстрактный базовый класс функций активации и их производных.
  + Linear – класс-наследник класса Function с реализованной линейной функцией активации.
  + Sigmoid – класс-наследник класса Function с реализованной сигмоидной функцией активации.
  + NeuralLink – класс, описывающий сущность нейронной связи.
  + NeuronFactory – базовый абстрактный класс для создания нейронов.
  + PerceptronNeuronFactory – класс-наследник класса NeuronFactory, создающий нейроны для персептрона.
  + NeuralNetwork – класс, содержащий реализацию нейронной сети.
  + TrainAlgorithm – базовый абстрактный класс для реализаций алгоритмов обучения сети.
  + Backpropagation – класс-наследник класса TrainAlgorithm, в котором реализован метод обратного распространения ошибки.
* NeuralNetwork\_train – консольное приложение для решения выбранной задачи, а именно для тренировки сети на тренировочных данных и сохранения метаданных сети.
* NeuralNetwork\_test – консольное приложение для тестирования обученной сети.

## Описание реализованных классов

Ниже приведена схема наследования реализованных классов (рис. 4).

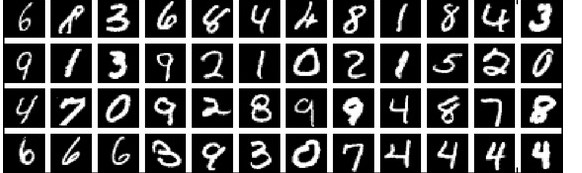


1. Схема наследования классов

# Результаты экспериментов

## Данные

В качестве решаемой задачи была выбрана одна из классических задач машинного обучения: детектирование рукописных цифр, представленных черно-белыми изображениями 28х28 пикселей. Данные взяты из базы данных MNIST [2].



1. Примеры изображений из базы данных MNIST [6]

Зная размеры каждого изображения, выделим наипростейшие признаки из них – пиксели. Таким образом, необходимо построить сеть, у которой 784 входа и 10 выходов, где каждый выход соответствует цифре, которую сеть выдает в качестве ответа.

Творческая часть задачи заключается в подборе количества и размеров скрытых слоев. Данная задача решена для различных конфигураций сетей, и для многих оценки качества выложены в сети Интернет.

Обучающая выборка состоит из 60000 изображений, представленных в виде строки, где первое число – правильный ответ, следующие за ним 784 числа – значения пикселов изображения.

## Показатели качества

Главным показателем качества будем считать контрольную ошибку. Для некоторой части данных, не входящих в обучающую выборку, подсчитаем количество верных и неверных откликов сети. За ошибку возьмем отношение неверных откликов ко всему размеру тестового набора. Соответственно, задача решена тем лучше, чем ниже контрольная ошибка.

## Результаты

Наименьшую контрольную ошибку удалось достичь при следующей архитектуре сети: полносвязная нейронная сеть с двумя скрытыми слоями по 256 нейронов. Использовался метод нормализации случайных весов при инициализации сети для более быстрого обучения. Темп обучения инициализирован постоянной величиной.

Обучающая выборка состояла из 20000 прецедентов и подавалась на вход алгоритму 15 раз, то есть проведено 15 эпох обучения. Алгоритм тестировался на 5000 объектах, не входящих в обучающую выборку.

При такой конфигурации сети контрольная ошибка составила 0,0604.

Ошибка для данных, входящих ранее в обучающую выборку, составила порядка 0,045.

Для сравнения использовалась библиотека машинного обучения TensorFlow [7] для языка Python. Написано приложение, создающее сеть аналогичной конфигурации и обучающее эту сеть аналогичным алгоритмом обратного распространения ошибки, реализованным в библиотеке TensorFlow. Контрольная ошибка составила 0,0601.

# Заключение

В ходе данной работы изучен один из метод обратного распространения ошибки для обучения нейронные сети. Разработана программная реализация, удовлетворяющая поставленным требованиям.

Проведен сравнительный анализ с существующей реализацией в библиотеке машинного обучения TensorFlow. Сравнение результатов контрольных ошибок в обеих реализациях позволяет сделать вывод о том, что алгоритм, реализованный в данной работе, работает корректно и со сравнимой эффективностью.

# Литература

1. Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин). — 141 с.
2. Хайкин С. Нейронные сети: Полный курс. Пер. с англ. Н. Н. Куссуль, А. Ю. Шелестова. 2-е изд., испр. — М.: Издательский дом Вильямс, 2008, 1103 с.
3. Laurene V. Fausett. Fundamentals of Neural Networks: Architectures, Algorithms And Applications: 1st (first) Edition Paperback — December 9, 1994 — 461 с.

# Ссылки

1. Репозиторий, содержащий разработанную программную реализацию [https://github.com/Maxim-Doronin/NeuralNetwork].
2. Данные, использованные для обучения [https://www.kaggle.com/c/digit-recognizer/data].
3. Лекция компании Яндекс по нейронным сетям на портале Хабрахабр [https://habrahabr.ru/company/yandex/blog/307260/].
4. Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин). — 141 с. — С. 106, Рис. 21.
5. Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин). — 141 с. — С. 107.
6. Ресурс, содержащий изображение данных обучающей выборки [https://www.researchgate.net/figure/264203784\_fig3\_Samples-from-the-MNIST-database].
7. Библиотека TensorFlow [https://www.tensorflow.org/].