Министерство образования Республики Беларусь

Учреждение образования  
«Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных сетей и систем

Кафедра электронный вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовой работе  
на тему

Система распознавания рукописных цифр

БГУИР КР 1-40 02 01 321 ПЗ

Студент С.И. Силюк

Руководитель А. М. Костенич

МИНСК 2017

Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Системное программное обеспечение вычислительных машин

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

\_\_\_\_\_\_\_\_\_ Д. И. Самаль

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

ЗАДАНИЕ

по курсовой работе студента

Силюка Сергея Ивановича

**1** Тема работы: «Система распознавания рукописных цифр»

**2** Срок сдачи студентом законченной работы: 24 мая 2017 г.

**3** Исходные данные к работе:

Язык программирования Python, интерактивная оболочка Jupyter Notebook. Реализовать программу для распознавания рукописных цифр. Программа должна основываться на нейронных сетях прямого распространения с обратным распространением ошибки. Работа с данными производится в интерактивной оболочке, где демонстрируется результаты наглядно в виде графиков и изображений. Данные для обучения и тестов хранятся в csv файле и включают: целевая переменная, 784 значений пикселей. В программе должна быть возможность просмотра статистики: количество ошибок для каждого класса на тестовой выборке.

**4** Содержание пояснительной записки (перечень подлежащих разработке вопросов):

Введение.

1. Обзор методов и алгоритмов решения поставленной задачи.

2. Обоснование выбранных методов и алгоритмов.

3. Описание программы для программиста.

4. Описание алгоритмов решения задачи.

5. Руководство пользователя.

6. Заключение.

7. Список литературы.

**5** Перечень графического материала:

**5.1** Структурная схема.

**5.2** Диаграмма - классов.

**5.3** Блок - схема.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов курсовой работы | Объём этапа, % | Срок выполнения этапа | Примечания |
| Разработка структурной схемы программы | 10 | 16.02–25.02 | С выполнением чертежей |
| Разработка диаграммы классов | 15 | 26.02–15.03 | С выполнением чертежей |
| Разработка основного функционала приложения с выполнением блок-схем | 50 | 16.03–15.04 | С выполнением чертежа |
| Разработка интерфейса | 15 | 16.04–01.05 |  |
| Завершение оформления пояснительной записки | 10 | 02.05–15.05 |  |

Дата выдачи задания: 15 февраля 2017 г.

Руководитель А. М. Костенич

ЗАДАНИЕ ПРИНЯЛ К ИСПОЛНЕНИЮ \_\_\_\_\_\_\_\_\_

**Содержание**

[**ВВЕДЕНИЕ 5**](#_Toc483876711)

[**1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ПОСТАВЛЕННОЙ ЗАДАЧИ 7**](#_Toc483876712)

[**2 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ 13**](#_Toc483876713)

[**3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА 16**](#_Toc483876714)

[**4 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 17**](#_Toc483876715)

[**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 18**](#_Toc483876716)

[**ЗАКЛЮЧЕНИЕ 22**](#_Toc483876717)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23**](#_Toc483876718)

[**ПРИЛОЖЕНИЕ А 24**](#_Toc483876719)

[**ПРИЛОЖЕНИЕ Б 25**](#_Toc483876720)

# **ВВЕДЕНИЕ**

Машинное обучение ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Machine Learning) — класс методов [искусственного интеллекта](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82), характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решению множества сходных задач. Для построения таких методов используются средства [математической статистики](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0), [численных методов](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B), [методов оптимизации](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), [теории вероятностей](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%B5%D0%B9), [теории графов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2), различные техники работы с [данными в цифровой форме](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)).

Общая постановка задачи для обучения. Имеется множество объектов (ситуаций) и множество возможных ответов (откликов, реакций). Существует некоторая зависимость между ответами и объектами, но она неизвестна. Известна только конечная совокупность прецедентов — пар «объект, ответ», называемая обучающей выборкой. На основе этих данных требуется восстановить зависимость, то есть построить алгоритм, способный для любого объекта выдать достаточно точный ответ. Для измерения точности ответов определённым образом вводится функционал качества.

Данная постановка является обобщением классических задач [аппроксимации](https://ru.wikipedia.org/wiki/%D0%90%D0%BF%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D1%8F) функций. В классических задачах аппроксимации объектами являются действительные числа или векторы. В реальных прикладных задачах входные данные об объектах могут быть неполными, неточными, нечисловыми, разнородными. Эти особенности приводят к большому разнообразию методов машинного обучения.

Так как раздел машинного обучения, с одной стороны, образовался в результате разделения науки о [нейросетях](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C) на методы обучения сетей и виды топологий архитектуры сетей, а с другой, вобрал в себя методы математической статистики, то указанные ниже способы машинного обучения исходят из нейросетей.

Искусственные нейронные сети применяются в различных областях науки: начиная от систем распознавания речи до распознавания вторичной структуры белка, классификации различных видов рака и генной инженерии. Однако, как они работают и чем они хороши?

Когда речь идет о задачах, отличных от обработки больших массивов информации, человеческий мозг обладает большим преимуществом по сравнению с компьютером. Человек может распознавать лица, даже если в помещении будет много посторонних объектов и плохое освещение. Мы легко понимаем незнакомцев даже когда находимся в шумном помещении. Но, несмотря на годы исследований, компьютеры все еще далеки от выполнения подобных задач на высоком уровне.

Человеческий мозг удивительно надежный: по сравнению с компьютером он не перестанет работать только потому, что несколько клеток погибнет, в то время как компьютер обычно не выдерживает каких-либо поломок в CPU. Но самой удивительной особенностью человеческого мозга является то, что он может учиться. Не нужно никакого программного обеспечения и никаких обновлений, если мы хотим научиться ездить на велосипеде.

Расчеты головного мозга производятся посредством тесно взаимосвязанных нейронных сетей, которые передают информацию, отсылая электрические импульсы через нейронные проводки, состоящие из аксонов, синапсов и дендритов. В 1943 году, компания McCulloch and Pitts смоделировала искусственный нейрон, как переключатель, который получает информацию от других нейронов и в зависимости от общего взвешенного входа, либо приводится в действие, либо остается неактивным. В узле ИНС пришедшие сигналы умножаются на соответствующие веса синапсов и суммируются. Эти коэффициенты могут быть как положительными (возбуждающими), так и отрицательными (тормозящими). В 1960 годах было доказано, что такие нейронные модели обладают свойствами, сходными с мозгом: они могут выполнять сложные операции распознавания образов, и они могут функционировать, даже если некоторые связи между нейронами разрушены. Демонстрация персептона Розенблатта показала, что простые сети из таких нейронов могут обучаться на примерах, известных в определенных областях.

Программа NETtalk применяла искусственные нейронные сети для машинного чтения текста и была первым широкоизвестным приложением. В биологии, точно такой же тип сети был применен для прогнозирования вторичной структуры белка; в самом деле, некоторые из лучших исследователей до сих пор пользуются тем же методом. С этого началась другая волна, вызвавшая интерес к исследованиям ИНС и поднявшая шумиху вокруг магического обучения мыслящих машин.

ИНС могут быть созданы путем имитации модели сетей нейронов на компьютере. Используя алгоритмы, которые имитируют процессы реальных нейронов, мы можем заставить сеть «учиться», что помогает решить множество различных проблем. Модель нейрона представляется как пороговая величина. Модель получает данные от ряда других внешних источников, определяет значение каждого входа и добавляет эти значения. Если общий вход выше пороговой величины, то выход блока равен единице, в противном случае – нулю. Таким образом, выход изменяется от 0 до 1, когда общая «взвешенная» сумма входов равна пороговой величине.

В представленной курсовой работе проведем исследование нейронной сети, обучим ее на наборе данных MNIST и сравним данный подход с другими методами машинного обучения.

# **1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ПОСТАВЛЕННОЙ ЗАДАЧИ**

Нейронные сети – это один из методов машинного обучения, к которому сейчас приковано достаточно большое внимание не только специалистов в области анализа данных или математиков, но и вообще людей, которые никак не связаны с этой профессией. И это связано с тем, что решения на основе нейронных сетей показывают самые лучшие результаты в самых различных областях человеческого знания, как распознавание речи, анализ текста, анализ изображений.

Базовая искусственная модель нейронной сети

Чтобы отразить суть биологических нейронных систем, определение искусственного [нейрона](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neuron) дается следующим образом:

Он получает входные сигналы (исходные данные либо выходные сигналы других нейронов [нейронной сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks)) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенную интенсивность (или вес); этот вес соответствует синаптической активности биологического нейрона. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина активации нейрона.

Сигнал активации преобразуется с помощью [функции активации](http://statsoft.ru/home/textbook/glossary/gloss_f.html#Activation Function) (или передаточной функции) и в результате получается выходной сигнал нейрона.

Если при этом использовать ступенчатую [функцию активации](http://statsoft.ru/home/textbook/glossary/gloss_f.html#Activation Function) (т.е., выход нейрона равен нулю, если вход отрицательный, и единице, если вход нулевой или положительный), то такой [нейрон](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neuron) будет работать точно так же, как описанный выше естественный нейрон (вычесть пороговое значение из взвешенной суммы и сравнить результат с нулем - это то же самое, что сравнить взвешенную сумму с пороговым значением). Учтите, что веса могут быть отрицательными, - это значит, что синапс оказывает на нейрон не возбуждающее, а тормозящее воздействие (в мозге присутствуют тормозящие нейроны).

Это было описание отдельного нейрона. Теперь возникает вопрос: как соединять нейроны друг с другом? Если сеть предполагается для чего-то использовать, то у нее должны быть входы (принимающие значения интересующих нас переменных из внешнего мира) и выходы (прогнозы или управляющие сигналы). Входы и выходы соответствуют сенсорным и двигательным нервам - например, соответственно, идущим от глаз и в руки. Кроме этого, однако, в сети может быть еще много промежуточных (скрытых) нейронов, выполняющих внутренние функции. Входные, скрытые и выходные нейроны должны быть связаны между собой.

Ключевой вопрос здесь - обратная связь (Haykin, 1994). Простейшая сеть имеет структуру [прямой передачи](http://statsoft.ru/home/textbook/glossary/gloss_p.html#Feedforward Networks) сигнала: Сигналы проходят от входов через скрытые элементы и в конце концов приходят на выходные элементы. Такая структура имеет устойчивое поведение. Если же сеть рекуррентная (т.е. содержит связи, ведущие назад от более дальних к более ближним нейронам), то она может быть неустойчива и иметь очень сложную динамику поведения. Рекуррентные сети представляют большой интерес для исследователей в области [нейронных сетей](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks), однако при решении практических задач, по крайней мере до сих пор, наиболее полезными оказались структуры прямой передачи, и именно такой тип нейронных сетей моделируется в нашей работе.

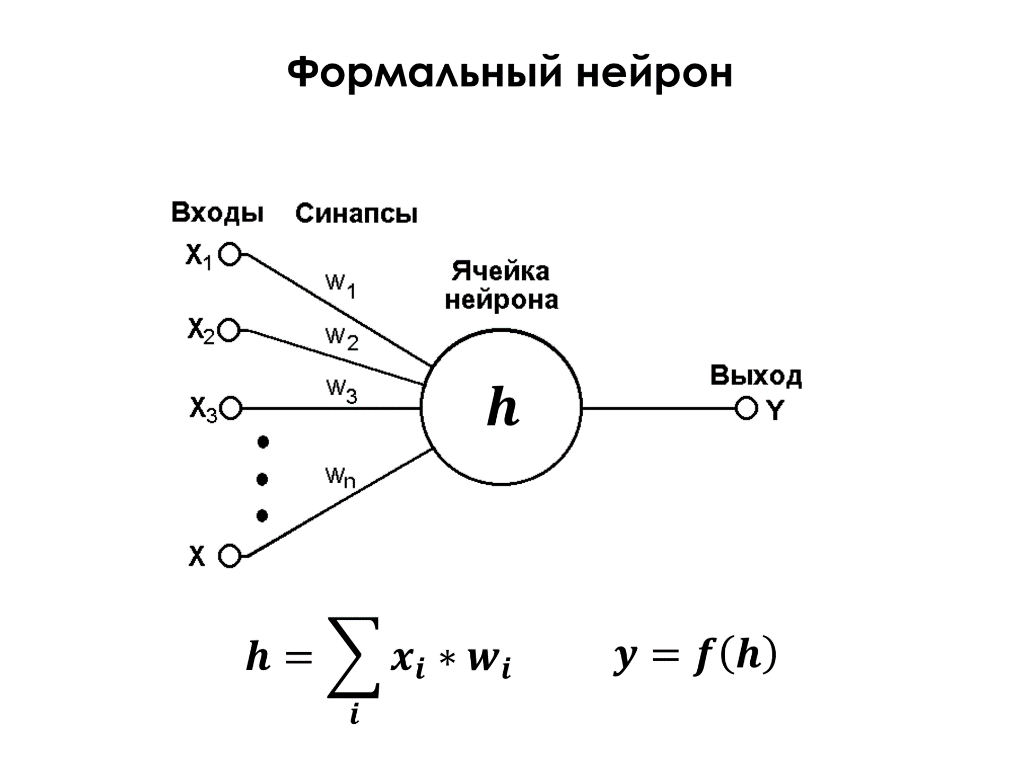


Рисунок 1.

С[еть с прямой передачей сигнала](http://statsoft.ru/home/textbook/glossary/gloss_p.html#Feedforward Networks), которую будем использовать, показана на рисунке. Нейроны регулярным образом организованы в слои. Входной слой служит просто для ввода значений входных переменных. Каждый из скрытых и выходных нейронов соединен со всеми элементами предыдущего слоя. Можно было бы рассматривать сети, в которых нейроны связаны только с некоторыми из нейронов предыдущего слоя; однако, для большинства приложений сети с полной системой связей предпочтительнее.

После того, как определено число слоев и число элементов в каждом из них, нужно найти значения для весов и порогов сети, которые бы минимизировали ошибку прогноза, выдаваемого сетью. Именно для этого служат алгоритмы обучения. С использованием собранных исторических данных веса и пороговые значения автоматически корректируются с целью минимизировать эту ошибку. По сути этот процесс представляет собой подгонку модели, которая реализуется сетью, к имеющимся обучающим данным. Ошибка для конкретной конфигурации сети определяется путем прогона через сеть всех имеющихся наблюдений и сравнения реально выдаваемых выходных значений с желаемыми (целевыми) значениями. Все такие разности суммируются в так называемую функцию ошибок, значение которой и есть ошибка сети. В качестве [функции ошибок](http://statsoft.ru/home/textbook/glossary/gloss_f.html#Error Function) чаще всего берется [сумма квадратов ошибок](http://statsoft.ru/home/textbook/glossary/gloss_k.html#Sum-squared error function), т.е. когда все ошибки выходных элементов для всех наблюдений возводятся в квадрат и затем суммируются.

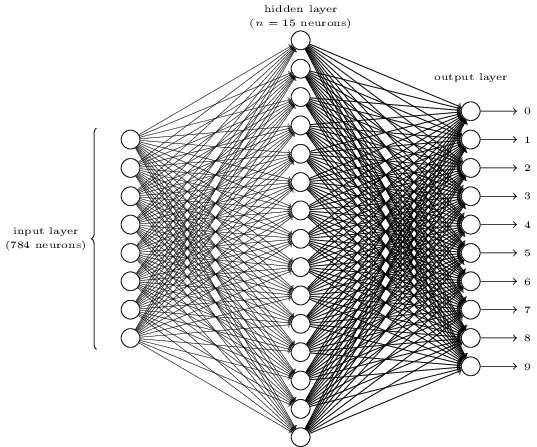


Рисунок 2.

Обучение многослойного персептрона

В традиционном моделировании (например, [линейном моделировании](http://statsoft.ru/home/textbook/glossary/gloss_l.html#Linear Modeling)) можно алгоритмически определить конфигурацию модели, дающую абсолютный минимум для указанной ошибки. Цена, которую приходится платить за более широкие (нелинейные) возможности моделирования с помощью [нейронных сетей](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks), состоит в том, что, корректируя сеть с целью минимизировать ошибку, мы никогда не можем быть уверены, что нельзя добиться еще меньшей ошибки.

В этих рассмотрениях оказывается очень полезным понятие поверхности ошибок. Каждому из весов и порогов сети (т.е. свободных параметров модели; их общее число обозначим через *N*) соответствует одно измерение в многомерном пространстве. N+1-е измерение соответствует ошибке сети. Для всевозможных сочетаний весов соответствующую ошибку сети можно изобразить точкой в N+1-мерном пространстве, и все такие точки образуют там некоторую поверхность - поверхность ошибок. Цель обучения нейронной сети состоит в том, чтобы найти на этой многомерной поверхности самую низкую точку.

В случае линейной модели с [суммой квадратов](http://statsoft.ru/home/textbook/glossary/gloss_k.html#Sum-squared error function) в качестве функции ошибок эта поверхность ошибок будет представлять собой параболоид (квадрику) - гладкую поверхность, похожую на часть поверхности сферы, с единственным минимумом. В такой ситуации локализовать этот минимум достаточно просто.

В случае нейронной сети поверхность ошибок имеет гораздо более сложное строение и обладает рядом неприятных свойств, в частности, может иметь [локальные минимумы](http://statsoft.ru/home/textbook/glossary/gloss_l.html#Local Minima) (точки, самые низкие в некоторой своей окрестности, но лежащие выше глобального минимума), плоские участки, седловые точки и длинные узкие овраги.

Аналитическими средствами невозможно определить положение глобального минимума на поверхности ошибок, поэтому обучение нейронной сети по сути дела заключается в исследовании поверхности ошибок. Отталкиваясь от случайной начальной конфигурации весов и порогов (т.е. случайно взятой точки на поверхности ошибок), алгоритм обучения постепенно отыскивает глобальный минимум. Как правило, для этого вычисляется градиент (наклон) поверхности ошибок в данной точке, а затем эта информация используется для продвижения вниз по склону. В конце концов алгоритм останавливается в нижней точке, которая может оказаться всего лишь локальным минимумом (а если повезет - глобальным минимумом).

Алгоритм обратного распространения

Самый известный вариант алгоритма обучения [нейронной сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks) - так называемый алгоритм [обратного распространения](http://statsoft.ru/home/textbook/glossary/gloss_o.html#Back Propagation) (back propagation; см. Patterson, 1996; Haykin, 1994; Fausett, 1994). Алгоритм обратного распространения наиболее прост для понимания, а в некоторых случаях он имеет определенные преимущества. Сейчас мы опишем его.

В алгоритме обратного распространения вычисляется вектор градиента поверхности ошибок. Этот вектор указывает направление кратчайшего спуска по поверхности из данной точки, поэтому если мы "немного" продвинемся по нему, ошибка уменьшится. Последовательность таких шагов (замедляющаяся по мере приближения к дну) в конце концов приведет к минимуму того или иного типа. Определенную трудность здесь представляет вопрос о том, какую нужно брать длину шагов.

При большой длине шага сходимость будет более быстрой, но имеется опасность перепрыгнуть через решение или (если поверхность ошибок имеет особо вычурную форму) уйти в неправильном направлении. Классическим примером такого явления при обучении [нейронной сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks) является ситуация, когда алгоритм очень медленно продвигается по узкому оврагу с крутыми склонами, прыгая с одной его стороны на другую. Напротив, при маленьком шаге, вероятно, будет схвачено верное направление, однако при этом потребуется очень много итераций. На практике величина шага берется пропорциональной крутизне склона (так что алгоритм замедляет ход вблизи минимума) с некоторой константой, которая называется [скоростью обучения](http://statsoft.ru/home/textbook/glossary/gloss_s.html#Learning Rate). Правильный выбор скорости обучения зависит от конкретной задачи и обычно осуществляется опытным путем; эта константа может также зависеть от времени, уменьшаясь по мере продвижения алгоритма.

Обычно этот алгоритм видоизменяется таким образом, чтобы включать слагаемое импульса (или инерции). Этот член способствует продвижению в фиксированном направлении, поэтому если было сделано несколько шагов в одном и том же направлении, то алгоритм "увеличивает скорость", что (иногда) позволяет избежать локального минимума, а также быстрее проходить плоские участки.

Таким образом, алгоритм действует итеративно, и его шаги принято называть [эпохами](http://statsoft.ru/home/textbook/glossary/gloss_ae.html#Epoch). На каждой эпохе на вход сети поочередно подаются все обучающие наблюдения, выходные значения сети сравниваются с целевыми значениями и вычисляется ошибка. Значение ошибки, а также градиента поверхности ошибок используется для корректировки весов, после чего все действия повторяются. Начальная конфигурация сети выбирается случайным образом, и процесс обучения прекращается либо, когда пройдено определенное количество эпох, либо когда ошибка достигнет некоторого определенного уровня малости, либо когда ошибка перестанет уменьшаться (пользователь может сам выбрать нужное [условие остановки](http://statsoft.ru/home/textbook/glossary/gloss_u.html#Stopping Conditions)).

Переобучение и обобщение

Одна из наиболее серьезных трудностей изложенного подхода заключается в том, что таким образом мы минимизируем не ту ошибку, которую на самом деле нужно минимизировать, - ошибку, которую можно ожидать от сети, когда ей будут подаваться совершенно новые наблюдения. Иначе говоря, мы хотели бы, чтобы нейронная сеть обладала способностью обобщать результат на новые наблюдения. В действительности сеть обучается минимизировать ошибку на обучающем множестве, и в отсутствие идеального и бесконечно большого обучающего множества это совсем не то же самое, что минимизировать "настоящую" ошибку на поверхности ошибок в заранее неизвестной модели явления (Bishop, 1995).

Сильнее всего это различие проявляется в проблеме [переобучения](http://statsoft.ru/home/textbook/glossary/gloss_p.html#Overlearning), или [слишком близкой подгонки](http://statsoft.ru/home/textbook/glossary/gloss_ch.html#Overfitting). Это явление проще будет продемонстрировать не для [нейронной сети](http://statsoft.ru/home/textbook/glossary/gloss_n.html#Neural Networks), а на примере аппроксимации посредством полиномов, - при этом суть явления абсолютно та же.

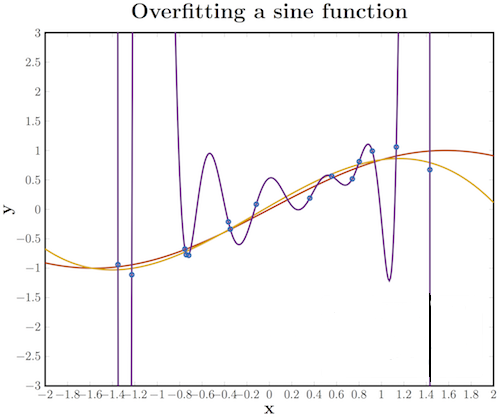


Рисунок 3.

# **2 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ И АЛГОРИТМОВ**

Рассмотрим работу двух наиболее важных методов: стохастический градиентный спуск и метод обратного распространения ошибки.

Градиентный спуск — [метод](http://dic.academic.ru/dic.nsf/ruwiki/877764) нахождения локального [минимума](http://dic.academic.ru/dic.nsf/ruwiki/193823) ([максимума](http://dic.academic.ru/dic.nsf/ruwiki/495482)) [функции](http://dic.academic.ru/dic.nsf/ruwiki/364163) с помощью движения вдоль [градиента](http://dic.academic.ru/dic.nsf/ruwiki/15964).

Используя как функцию стоимости средне квадратичную ошибку, гиперплоскость данной функции будет выглядеть в виде парабалоида.

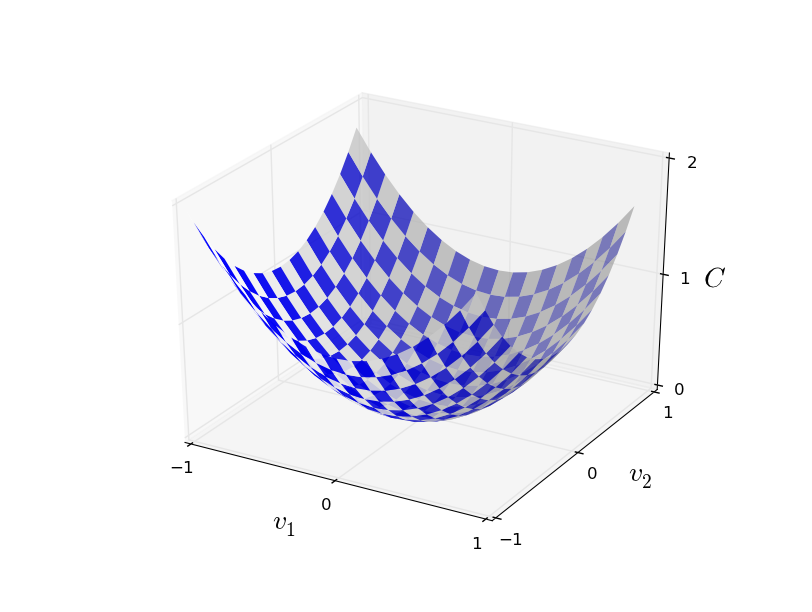


Рисунок 4.

Как известно, производная функции в точке показывает направление возрастания функции. Отрицательное значение производной укажет направление движения к минимуму.

Вычисляя производную и двигаясь вдоль нее, мы можем достигнуть минимума функции ошибки.

Иллюстрация последовательных приближений к точке экстремума в направлении наискорейшего спуска(красн.) в случае дробного шага. Синим отмечены линии уровня.

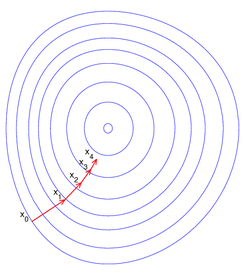


Рисунок 5.

Обозначим вектор градиента следующим образом:

Тогда уравнение примет вид:

Важным настраиваемым параметром градиентного спуска является η, скорость, с которой будем передвигаться. Если взять этот параметр слишком большой, то мы можем перепрыгнуть через минимум, если слишком малый, то потребуется большое количество итераций, так же мы можем попасть в локальный минимум.

Переходя к функции корректировки весов и смещений, запишем уравнения:

Однако, вычислять производные для каждой выборки слишком затратно. Поэтому используют метод стохастического градиентного спуска, где случайно берется выборка фиксированного размера и вычисляется ошибка для нее. Т.е.:

Метод обратного распространения ошибки основывается на четырех уравнения. Выведем или докажем некоторые из них.

Будем использовать следующие обозначения:

– вес связи k-го нейрона (l-1)-го слоя и j-го нейрона l-го слоя,

– смещение j-го нейрона l-го слоя,

– активация j-го нейрона l-го слоя.

По определению:

Применяя цепочку правил, перепишем данное уравнение следующим образом в частных производных:

Функция активации выходного слоя имеет смысл только тогда, когда j=k. Известно, что , тогда первое уравнение для ошибки выходного слоя будет иметь следующий вид:

(1)

Следующее уравнение определяет ошибку слоя , используя ошибку следующего слоя

Дифференцируя последнее уравнение по , мы получим:

Подставляя в уравнение, которое находится выше, получим второе уравнение для ошибки в слое:

(2)

Действуя по такому же принципу, можно получить уравнения (3) и (4):

# **3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА**

Код программы, основном, заключен в классе Network.

Поля данного класса:

* num\_layers – количество слоев;
* sizes – список, содержащий размеры каждого слоя;
* biases – матрица смещений нейронов;
* weights – матрица весов нейронов.

Методы класса Network:

* \_\_init\_\_(self, sizes) – конструктор класса.
* feedforward(self, a) – функция, которая возвращает выход из нейронной сети, если на вход подано a, список вероятностей обнаружения цифры.
* SGD(self, training\_data, epochs, mini\_batch\_size, eta, test\_data = None) – функция обучения нейронной сети с помощью стохастического градиентного спуска, если задать параметр test\_data, то после каждой эпохи будет отображаться частичный прогресс.
* update\_mini\_batch(self, mini\_batch, eta) – функция обновлений весов и смещений сети путем применения градиентного спуска с использованием обратного распространения ошибки.
* backprop(self, x, y) – функция, вычисляющая градиент для весов и смещений каждого слоя.
* evaluate(self, test\_data) – функция, возвращающая количество правильных предсказанных нейронной сетью значений на тестовой выборке.
* cost\_derivative(self, output\_activations, y) – функция, вычисляющая вектор частных производных.

# **4 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ**

Рассмотрим несколько алгоритмов по шагам.

Алгоритм backpropagation: Начало.

1. Input x: установить значения активации для входного слоя.
2. Feedforward: для каждого l=2,3,...,L вычисляется и .
3. Output error : вычисляется выходной вектор ошибок .
4. Backpropagate the error: для каждого l = L-1, L-2, ... , 2 вычисляется .
5. Output: градиент функции стомости для весов и смещений и .
6. Конец.

На практике комбинируют алгоритм обратного распространения ошибки с алгоритмом обучения стохастическим градиентным спуском.

1. Устанавливаем выборку для обучения.
2. Для каждого значения выборки устанавливаем значение активации и выполняем шаги 2,3,4 алгоритма backpropagation.
3. Градиентный спуск: для каждого L = L, L-1, ... , 2 обновляем веса в соответствии с правилом:
4. Конец.

# **5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

В данном разделе мы рассмотрим применение полученной нейронной сети и рассмотрим зависимость точности оценки от ее параметров, разберемся к чему это может привести. Все графики построены при помощи библиотеки Matplotlib.

Первый параметр - это количество нейронов во внутреннем слое.

График данной зависимости представлен на рисунке 6.

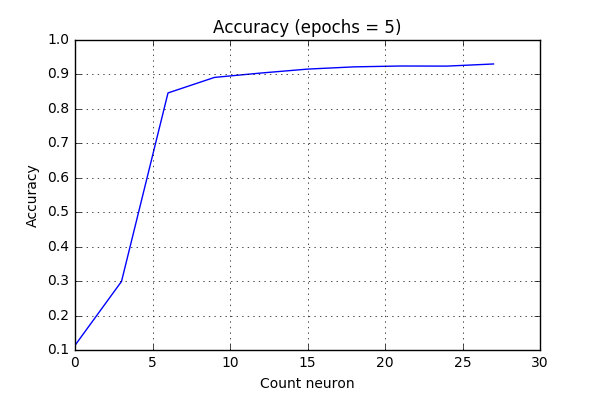


Рисунок 6.

Как видно из графика, точность нейронной сети возрастает с ростом количества нейронов во внутреннем слое, однако возрастает и время обучения нейронной сети. Как видно, при значениях больше 15 точность сети становится стабильной.

Второй параметр – это количество эпох, во время которых мы корректируем веса и смещения. График представлен на рисунке 7.

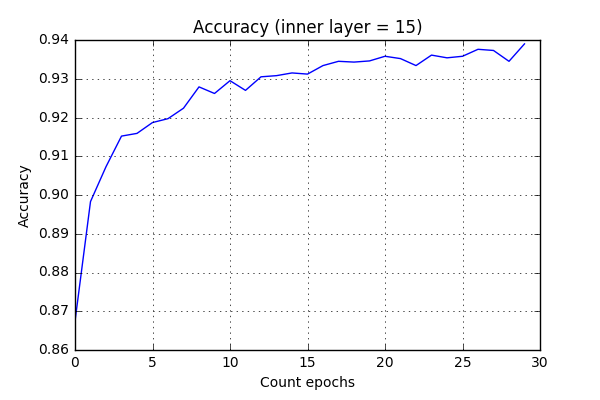


Рисунок 7.

Из графика видно, что точность возрастает с увлечением количества эпох. При значениях больше 15 - 20 точность становится стабильной с определенной ошибкой. Эту ошибку можно объяснить тем, что стохастический градиентный спуск использует случайную выборку для обновления весов и смещений, поэтому точно сказать, как будут обновляться веса и смещения мы не можем. Также при увеличении эпох мы можем столкнуться с проблемой как переобучения, так как алгоритм будет подстраиваться под обучающую выборку.

Третий параметр – это размер эпохи. График представлен на рисунке 8.

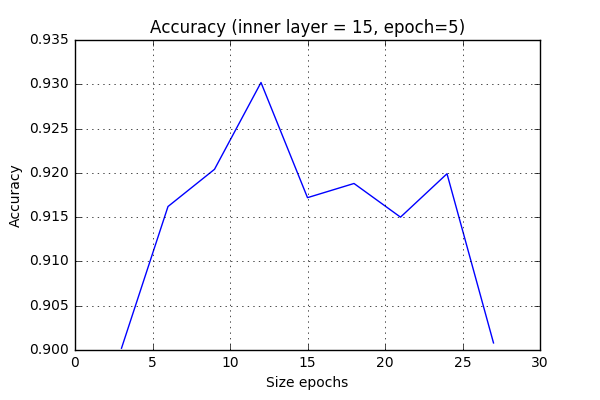


Рисунок 8.

Как видно из графика, данный параметр довольно нестабилен, что объясняется случайностью выбора мини-пакета при градиентном спуске. Данный параметр при малых значениях будет приводить к резким изменениям весов и смещений, а при больших значениях затраты будут во времени на вычисление градиента для каждого набора.

Четвертый параметр – это скорость или шаг градиентного спуска. График представлен на рисунке 9.

Из графика видно, что при значениях больше 3, точность становится стабильной. Размер шага определяет на сколько мы сдвинемся в направлении экстремумы функции ошибки. Однако при малых значениях мы очень долго будем идти к минимуму или достигнем локального минимума при больших потерях во времени. При слишком больших значениях мы можем пропустить, пройти глобальный минимум, или же мы будет будем перемещаться из стороны в стороны, что за собой влечет большое значение дисперсии, т.е. большой разброс значений.

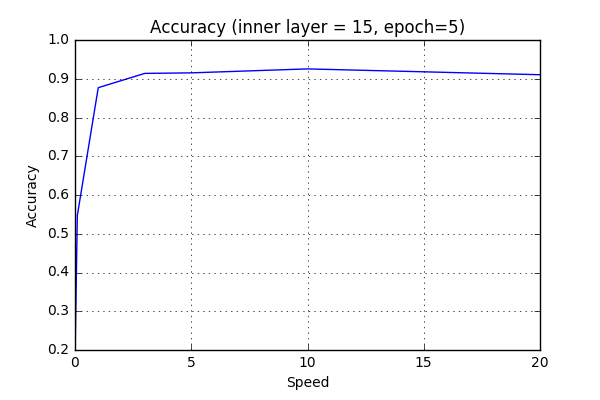


Рисунок 9.

Чтобы убедиться в корректности работы нейронной сети, приведем гистограмму ошибок на тестовой выборке для каждой цифры.



Рисунок 10.

Так как размер тестовой выборки 10000, то можно сказать, что ошибки для каждой цифры распределены равномерно, с небольшой дисперсией. В некоторых случаях можно столкнуться с проблемой, что ошибка на одной цифре превышает в несколько раз ошибки на других цифрах. С такой проблемой можно бороться посредством пересмотра выборки обучающейся и тестовой.

Так же, чтобы убедиться, что сеть не переобучена, можно просмотреть точность как на обучающейся выборке, так и на тестовой, если значения сильно различаются, то это говорит о том, что сеть переобучена или недообучена.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы была разработана нейронная сеть прямого распространения, а также протестирована на наборе данных MNIST, т.е. на наборе рукописных цифр. В дальнейшем для увеличения результата точности можно протестировать другие функции активации, а также использовать сверточные нейронные сети.

Нейронные сети имеют широкое применение в нашем современном мире. Мы можем обучить нейронную сеть на определенном наборе данных, а в программу подставить лишь значения весов и смещений.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

Лутц, М. Изучаем Python / М. Лутц. – СПб.: Символ-Плюс, 2011.

[Электронный ресурс]: <http://neuralnetworksanddeeplearning.com>

[Электронный ресурс]: <https://habrahabr.ru/post/134998/>

[Электронный ресурс]: http://statsoft.ru/home/textbook/modules/stneunet.html

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Схема структурная

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Диаграмма классов