

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

(наименование факультета)

Кафедра «ПОВТиАС»

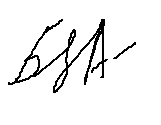
(наименование кафедры)

**ОТЧЕТ**

по учебной ознакомительной практике

вид практики

на ФГБОУ ВО «Донской государственный технический университет»

 наименование базы практики

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бережнов А.С

подпись, датаИ.О.Ф.

Обозначение отчета УП.78.0000.000 Группа Впр-13

Направление подготовки (специальности)

09.03.04 Программная инженерия

коднаименование направления подготовки

Направленность (Профиль) Программное обеспечение вычислительной техники и автоматизированных систем

Руководитель практики:

от кафедры доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Скляренко

должность подпись, дата имя, отчество, фамилия

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата подпись преподавателя

Ростов-на-Дону

2020 г.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

(наименование факультета)

Кафедра «ПОВТиАС»

(наименование кафедры)

**ЗАДАНИЕ**

по учебной ознакомительной практике

вид практики

на ФГБОУ ВО «Донской государственный технический университет»

наименование базы практики

кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

в период с «10» \_\_\_\_февраля\_\_ 2020 г. по «\_6\_» \_\_\_июня\_\_\_ 2020 г.

Обучающийся Бережнов Артём Сергеевич

Обозначение отчета УП.78.0000.000 Группа Впр-13

|  |  |  |
| --- | --- | --- |
| Руководитель практики от  кафедры | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | А.А. Скляренко  имя, отчество, фамилия |
|  | C:\Users\Артем\Desktop\Новый точечный рисунок (2).bmp |  |
| Задание принял к исполнению | C:\Users\Артем\Desktop\Новый точечный рисунок (2).bmp\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | Бережнов А.С  имя, отчество, фамилия |

Срок представления отчета на кафедру «\_6\_» \_\_\_июня\_\_\_\_\_ 2020 г.

Содержание индивидуального задания:

Разработать программное средство аппроксимации тригонометрических функций синуса и косинуса нейронной сетью.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

(наименование факультета)

Кафедра «ПОВТиАС»

(наименование кафедры)

**Рабочий график (план) проведения практики**

|  |  |  |
| --- | --- | --- |
| **№** | **Мероприятие** | **Срок выполнения** |
| 1 | Прохождение вводного и первичного инструктажа по охране труда на рабочем месте, и инструктажа по пожарной безопасности на объекте. | 10.02.2020 |
| 2 | Аналитический обзор | с 11.02.2020 по 5.04.2020 |
| 3 | Алгоритмическое конструирование | с 6.04.2020 по 19.04.202 |
| 4 | Программная реализация | с 20.04.2020 по 17.05.2020 |
| 5 | Тестирование приложения | с 18.05.2020 по 20.05.2020 |
| 6 | Подготовка итогового отчета | с 21.05.2020 по 5.06.2020 |
| 7 | Защита итогового отчета | 6.06.2020 |

Руководитель практики:

от кафедры \_\_\_ доцент\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Скляренко  должность подпись, дата имя, отчество, фамилия

Ростов-на-Дону

2020 г.

ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Место работы | Выполняемые работы | Оценка руководителя |
| 10.02.2020 –23.02.2020 | Кафедра «ПОВТиАС» | Прохождение вводного и первичного инструктажа по охране труда на рабочем месте. Прохождение инструктажа по пожарной безопасности на объекте. Разбор организационных требований и методических рекомендаций к оформлению итогового отчета и порядку его защиты |  |
| 24.02.2020 –15.03.2020 | Кафедра «ПОВТиАС» | Проведение аналитического обзора метода аппроксимации функции нейронной сетью и постановка задачи |  |
| 16.03.2020 –1.04.2020 | Кафедра «ПОВТиАС» | Написание первой главы «Аналитический обзор метода аппроксимации функции нейронной сетью» итогового отчета |  |
| 2.04.2020 –5.04.2020 | Кафедра «ПОВТиАС» | Изучение документации и составление технического задания (приложение А итогового отчета) |  |
| 6.04.2020 –16.04.2020 | Кафедра «ПОВТиАС» | Алгоритмическое конструирование:  · Общий алгоритм работы  · Алгоритм обучения сети  · Алгоритм работы сети |  |
| 17.04.2020 –19.04.2020 | Кафедра «ПОВТиАС» | Написание второй главы «Алгоритмическое конструирование» |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 20.04.2020 –10.05.2020 | Кафедра «ПОВТиАС» | Программная реализация  · Выбор языка программирования и среды разработки  · Реализация базовых алгоритмов  · Реализация интуитивно-понятного интерфейса |  |
| 11.05.2020 –17.05.2020 | Кафедра «ПОВТиАС» | Тестирование программного средства и его корректировка |  |
| 18.05.2020 – 20.05.2020 | Кафедра «ПОВТиАС» | Сдача программного средства |  |
| 21.05.2020 –23.05.2020 | Кафедра «ПОВТиАС» | Написание третей главы «Программное конструирование» |  |
| 24.05.2020 –26.05.2020 | Кафедра «ПОВТиАС» | Написание четвертой главы «Тестирование программного средства» |  |
| 27.05.2020 –31.05.2020 | Кафедра «ПОВТиАС» | Подготовка итогового отчета |  |
| 1.06.2020 –6.06.2020 | Кафедра «ПОВТиАС» | Защита итогового отчета |  |

Руководитель практики:

от кафедры \_\_\_\_ доцент\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ А.А. Скляренко\_\_\_\_\_\_\_\_\_

должность подпись, дата имя, отчество, фамилия

ОТЗЫВ – ХАРАКТЕРИСТИКА

Обучающийся Бережнов Артем Сергеевич

фамилия, имя, отчество

1 курса группы ВПР13 кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Вид практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная ознакомительная\_\_\_\_\_\_\_\_\_\_\_

Наименование места практики кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

наименование предприятия, структурного подразделения

Обучающийся выполнил задания программы практики

\_\_разработал программное средство аппроксимации тригонометрических функций синуса и косинуса нейронной сетью.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дополнительно ознакомился/изучил

Более детально познакомился с языком программирования С++, изучил принцип работы простейших нейронных сетей. Принцип реализации текстового интерфейса консольного приложения.\_\_\_\_\_\_\_\_\_\_\_\_

Заслуживает оценки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | Руководитель практики  от кафедры  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_ г. |

**Содержание**

[ВВЕДЕНИЕ 8](#_Toc77186289)

[1. Аналитический обзор метода аппроксимации функций нейронной сетью 9](#_Toc77186290)

[1.1 Общее понятие нейронных сетей. 9](#_Toc77186291)

[1.2 Обзор задачи аппроксимации 13](#_Toc77186292)

[1.3 Постановка задачи 15](#_Toc77186293)

[1.4 Выводы по главе 16](#_Toc77186294)

[2. Алгоритмическое конструирование 17](#_Toc77186295)

[2.1 Общий алгоритм работы программного средства 17](#_Toc77186296)

[2.2 Алгоритм обучения нейронной сети 17](#_Toc77186297)

[2.3 Алгоритм функционирования нейронной сети 19](#_Toc77186298)

[2.4 Вывод 20](#_Toc77186299)

[3. Программное конструирование 21](#_Toc77186300)

[3.1 Выбор языка программирования 21](#_Toc77186301)

[3.2 Выбор среды программирования 22](#_Toc77186302)

[3.3 Описание основных функций 22](#_Toc77186303)

[3.3.1 Используемые функции 22](#_Toc77186304)

[3.3.2 Реализация обучения сети 25](#_Toc77186305)

[3.3.3 Реализация работы сети 28](#_Toc77186306)

[3.4 Выводы по главе 29](#_Toc77186307)

[4. Тестирование программного средства 30](#_Toc77186308)

[4.1 Контрольные примеры обучения и работы нейронной сети 30](#_Toc77186309)

[4.2 Результаты тестирования программного средства 30](#_Toc77186310)

[4.3 Выводы по главе 32](#_Toc77186311)

[ЗАКЛЮЧЕНИЕ 33](#_Toc77186312)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 34](#_Toc77186313)

[ПРИЛОЖЕНИЕ А Техническое задание 35](#_Toc77186314)

[ПРИЛОЖЕНИЕ Б Исходный код программного средства 41](#_Toc77186315)

# **Введение**

Исследования в области нейронных сетей начались в 40-е годы XX века. Первое систематическое изучение искусственных нейронных сетей было предпринято Маккалокком и Питтсом в 1943 г. Позднее они исследовали сетевые парадигмы для распознавания изображений, подвергаемых сдвигам и поворотам.

Основные задачи, которые ставятся перед нейронными сетями, относятся к задачам распознавания образов. Они заключаются в том, чтобы классифицировать входной образ, то есть отнести его к какому-либо известному сети классу. Изначально сети даются эталонные образы – такие образы, принадлежность которых к определенному классу известна. Затем на вход сети подается некоторый неизвестный образ, и сеть пытается по определенному алгоритму соотнести его с каким-либо эталонный образом. Можно сказать, что нейросети проводят кластеризацию образов. Так как кластерный анализ применяется исследователями рынка ценных бумаг, то нейронные сети могут быть использованы и для прогнозирования стоимости акций, что является актуальной задачей, к тому же строго неразрешимой на данный момент.

Так же нейронный сети могут применяться в криминалистике (анализ отпечатков пальцев) или же для облегчения работы правоохранительных органов в поимке преступников, потому что сейчас уже созданы такие программы, которые распознают лица. В первой главе приведен обзор предметной области аппроксимации функций нейронной сетью, а также сформулировано их условие.

# **Аналитический обзор метода аппроксимации функций нейронной сетью**

В данной главе рассматривается понятие нейрона, нейронный сетей, аппроксимации математических функций, подробно описываются методы аппроксимации функций нейронной сетью, определяется цель работы и задачи для ее достижения.

## **Общее понятие нейронных сетей.**

Искусственная нейронная сеть (ИНС) представляет собой математическую модель человеческого мозга. Как известно, мозг состоит из множества частиц – нейронов, соединенных между собой связями, передающими сигналы. Проходя через нейроны, сигналы изменяются. Наличие множества таких нейронов позволяет человеку решать сложные задачи, запоминать большие объемы информации и принимать решения. Создание такой модели позволило научить компьютер некоторым произвольным действиям, например сложению чисел. Для обучения нейронной сети предъявляется обучающая выборка, и сеть самостоятельно подстраивает свою структуру для оптимального моделирования данной выборки. После обучения сеть оказывается способной вычислять результат, имея на выходе данные, отсутствующие в обучающей выборке.

Биологический нейрон (Cell) имеет ядро (Nucleus), а также отростки нервных волокон двух типов - дендриты (Dendrites), по которым принимаются импульсы (Carries signals in), и единственный аксон (Axon), по которому нейрон может передавать импульс (Carries signals away). Аксон контактирует с дендритами других нейронов через специальные образования – синапсы (Synapses), которые влияют на силу передаваемого импульса. Структура, состоящая из совокупности большого количества таких нейронов, получила название биологической (или естественной) нейронной сети.

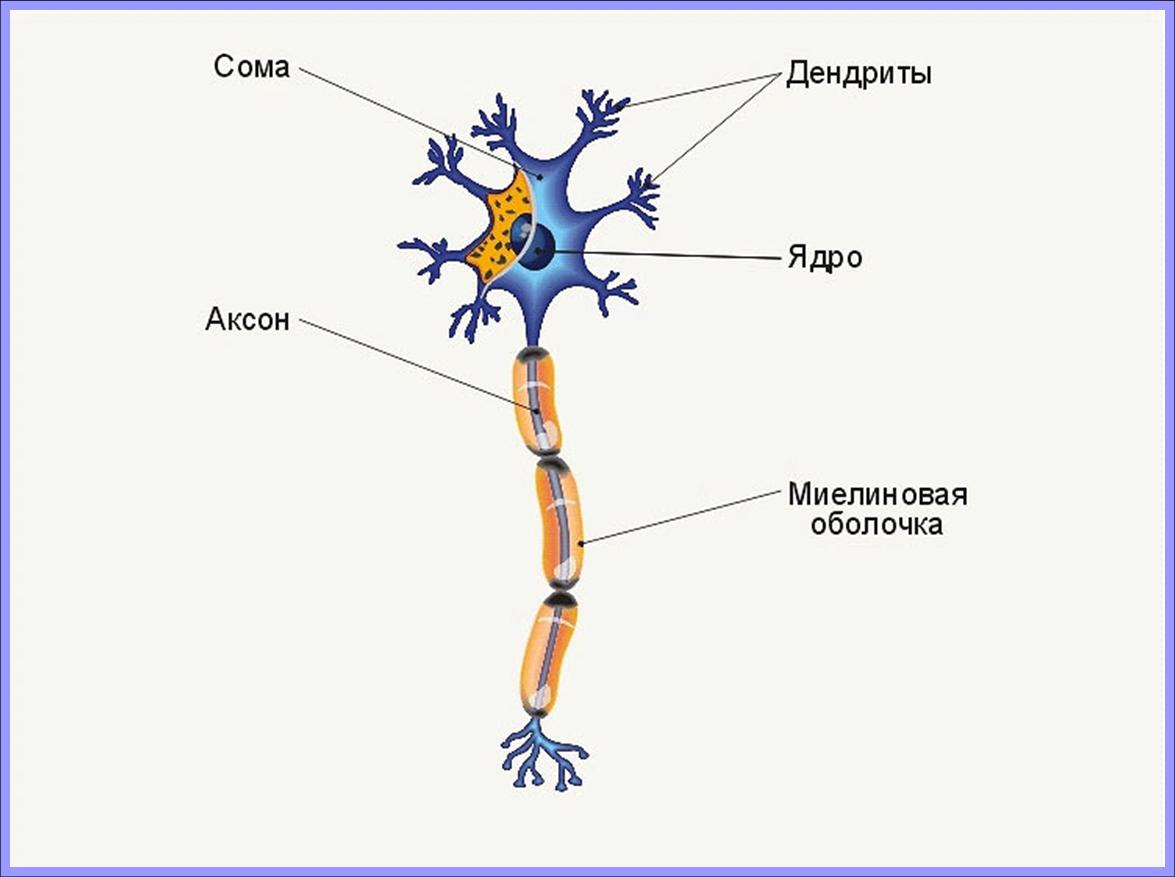


Рисунок 1.1 - Нейрон

Появление формального нейрона во многом обусловлено изучением биологических нейронов. Формальный нейрон является основой любой искусственной нейронной сети. Нейроны представляют собой относительно простые, однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены и заторможены. Искусственный нейрон, так же как и его естественный прототип, имеет группу синапсов (входов), которые соединены с выходами других нейронов, а также аксон – выходную связь данного нейрона, откуда сигнал возбуждения или торможения поступает на синапсы других нейронов.

(ко­то­ры­ми, напр., яв­ля­ют­ся вы­ход­ные сиг­на­лы др. фор­маль­ных ней­ро­нов Н. с.) в вы­ход­ной сигнал, пред­став­лен на рис. 1.1.

Здесь yy – зна­че­ние вы­хо­да фор­маль­но­го ней­ро­на; aiai – ве­со­вые ко­эф­фи­ци­ен­ты; xixi – вход­ные зна­че­ния фор­маль­но­го ней­ро­на (xi∈{0,1},x0=1xi∈{0,1},x0=1). Про­цесс вы­чис­ле­ния вы­ход­но­го зна­че­ния фор­маль­но­го ней­ро­на пред­став­ля­ет со­бой дви­же­ние по­то­ка дан­ных и их пре­об­ра­зо­ва­ние. Сна­ча­ла дан­ные по­сту­па­ют на блок вхо­да фор­маль­но­го ней­ро­на, где про­ис­хо­дит ум­но­же­ние ис­ход­ных дан­ных на со­от­вет­ст­вую­щие ве­со­вые ко­эф­фи­ци­ен­ты, т. н. синоптические веса (в соответствии с синапсами биологических нейронов). Ве­со­вой ко­эф­фи­ци­ент яв­ля­ет­ся ме­рой, ко­то­рая оп­ре­де­ля­ет, на­сколь­ко со­от­вет­ст­вую­щее вход­ное зна­че­ние влия­ет на со­стоя­ние фор­маль­но­го ней­ро­на. Ве­со­вые ко­эф­фи­ци­ен­ты мо­гут из­ме­нять­ся в со­от­вет­ст­вии с обу­чаю­щи­ми при­мера­ми, ар­хи­тек­ту­рой Н. с., пра­ви­ла­ми обу­че­ния и др.

По­лу­чен­ные (при ум­но­же­нии) зна­че­ния пре­об­ра­зу­ют­ся в сум­ма­то­ре в од­но чи­сло­вое зна­че­ние gg (по­сред­ст­вом сум­ми­ро­ва­ния). За­тем для оп­ре­де­ле­ния выхо­да фор­маль­но­го ней­ро­на в бло­ке не­ли­ней­но­го пре­об­ра­зо­ва­ния (реа­ли­зую­ще­го пе­ре­да­точ­ную функ­цию) gg срав­ни­ва­ет­ся с не­ко­то­рым чис­лом (по­ро­гом). Ес­ли сум­ма боль­ше зна­че­ния по­ро­га, фор­маль­ный ней­рон ге­не­ри­ру­ет сиг­нал, в про­тив­ном слу­чае сиг­нал бу­дет ну­ле­вым или тор­мо­зя­щим.

Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Теоретически число слоёв и число нейронов в каждом слое нейронной сети может быть произвольным, однако фактически оно ограничено ресурсами компьютера или специализированной микросхемы, на которых обычно реализуется нейронная сеть. При этом если в качестве активационной функции для всех нейронов сети используется функция единичного скачка, нейронная сеть называется многослойным персептроном.

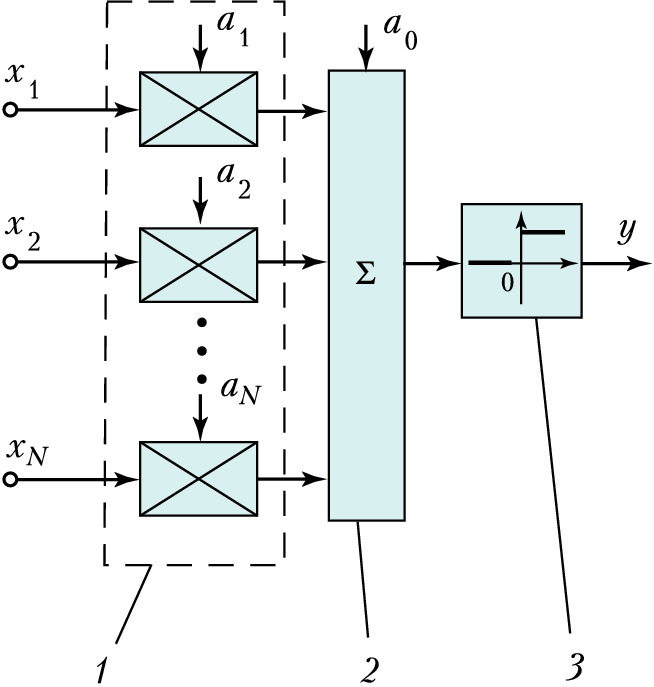


Рисунок 1.2 – Многослойный персептрон

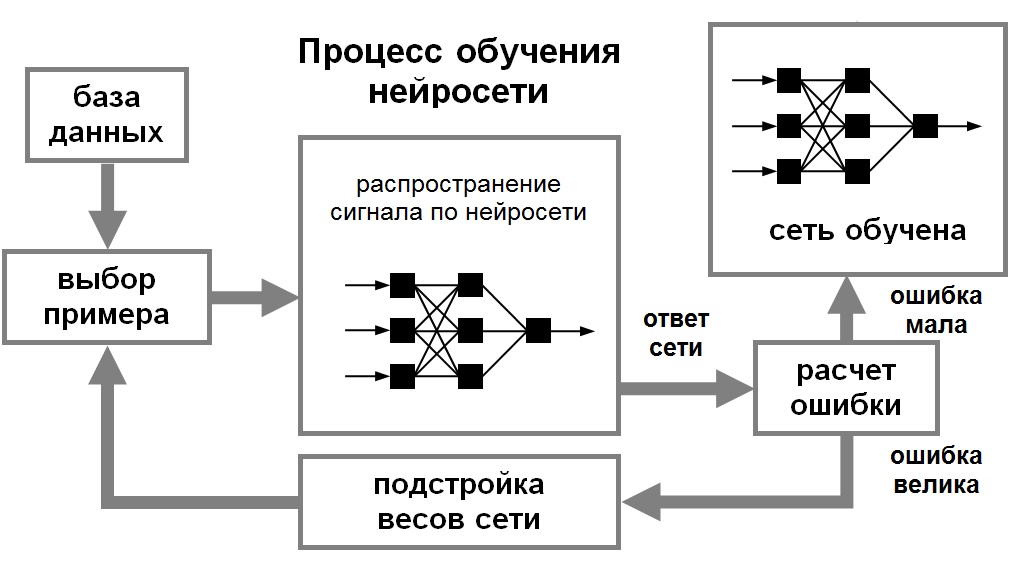


Рисунок 1.3 – Процесс обучения нейросети

## **Обзор задачи аппроксимации**

Аппроксимация (от лат. Приближаюсь) – замена одних математических объектов другими в том или ином смысле близкими к исходным. Аппроксимация позволяет исследовать числовые характеристики или качественные свойства объекта сводя задачу к изучению более простых или удобных объектов, например таких характеристики которых легко вычисляются или свойства которых уже известны. Приближение – то же, что аппроксимация, термин приближение иногда употребляется в смысле приближающегося объекта. Приближение функций – нахождение для данной функции, функции из некоторого определенного класса, например среди алгебраических многочленов заданной степени, в том или ином смысле близкой к дающей ее приближенное представление. [2]

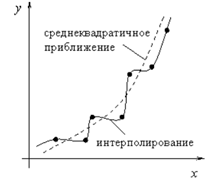
 Аппроксимация, при которой приближение строится на заданом дискретном множестве точек {xj}, называется точечной.

Рисунок 1.4 – График аппрокисмации функции

Для получения точечного среднеквадратичного приближения функции y=f(x), заданной таблично, аппроксимирующую функцию Ф(x) строят из условия минимума величины.

(1.1)

Задача аппроксимации – представление произвольной сложной функции f(x) более простой и удобной для практического использования аппроксимирующей функцией j(x) таким образом, чтобы отклонение j (x) от f(x) на заданном отрезке [a,b] было минимальным по определенному критерию приближения. При этом в отличие от задачи интерполяции значения функции j(x) могут отличаться от значений функции f(x) в заданных точках. Наиболее распространенным методом аппроксимации данных является метод наименьших квадратов. Критерий близости в методе наименьших квадратов является требование минимальности среднего квадратического отклонения.

(1.2)

Из всех прямых ϕ (x) = ax + b мы выбираем ту, для которой сумма

квадратов отклонений значений функции от этой прямой минимальна. [3].

## **Постановка задачи**

Проведенный обзор возможностей нейронных сетей показал эффективность их применения для аппроксимации функций. Таким образом целью учебно-ознакомительной практики является разработка программного средства аппроксимации тригонометрических функций синуса и косинуса нейронной сетью.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать алгоритмы аппроксимации функций косинуса и синуса нейронной сетью;
* выбрать среду и язык реализации;
* реализовать подпрограммы: нахождения значений весов, обучения сети, записи весов в файл и др.;
* разработать модуль интуитивно-понятного интерфейса программного средства (ПС) и выполнить его интеграцию с написанными подпрограммами;
* провести тестирование ПС на нескольких контрольных примерах;
* в соответствии с отчетом по тестированию выполнить корректировку ПС;
* разработать сопроводительную документацию: техническое задание, отчет.;

Разрабатываемое программное средство будет обладать следующими функциями:

* Записи весов в файл
* Обучения сети
* Работы сети

## **Выводы по главе**

В первой главе было рассмотрено общее понятие нейронных сетей, описана задача аппроксимации и подробно изучен метод аппроксимации нейронной сетью.

# **Алгоритмическое конструирование**

В данном разделе рассматриваются общий алгоритм работы программного средства, алгоритм обучения нейронной сети, алгоритм работы нейронной сети. Приведено их описание и схема работы.

## **Общий алгоритм работы программного средства**

Перед пользователем открывается консольное окно, в котором необходимо выбрать либо обучение сети, либо работа с уже обученной сетью, при выборе обучения сети пользователю предоставляется выбор либо обучать сеть из обучающего файла, либо вводить значения для обучения вручную. После выбора пользователь выбирает название файла куда будут записываться в дальнейшем измененные веса, далее изначальные веса заполняются случайными числами от 0 до 1. После чего происходит непосредственно сам алгоритм обучения сети. Результат т.е. измененные веса записываются в файл, заданный пользователем. После обучения пользователю предоставляется возможность еще раз обучить сеть или же уже работать с сетью. В целом процесс обучения и работы зациклен, поэтому после одного можно перейти к другому. Если же изначально был сделан выбор с работой программы, то в силу приходит алгоритм по работе. Аналогичным образом, после обучения пользователь может перейти в обучение и обратно. По результату работы пользователь получает либо желаемый результат работы сети, либо обученную сеть.

## **Алгоритм обучения нейронной сети**

Алгоритм обучения сети представляет из себя полный расчет работы сети по каждому из 3 нейронов, с изначальными весами которые задаются случайным образом, после этого значения весов изменяются при помощи метода обратного распространения ошибки. После обучения сеть выдает значение работы сети с уже измененными весам. Измененные веса записываются в файл. Заданный пользователем, с заданным им же именем.

Начало

А

Цикл i от 0 до 3

Цикл i от 0 до 200

Цикл j от 0 до 2

work1(y1, x1, w, f1, F1)

work2(y2, x1, w, f2, F2)

work3(y3, x1, w, f3, F3) work(Y, F1, F2, F3, Y1) education(a, x1, Y1, d, t, Y, d1, d2, d3, d4, d5, d6, f1, f2, f3, w) write(w, name)

w[i][j] = (rand() % 70) \* 0.01

Увеличить цикл j на 1

Вывод("Ответ сети ", Y1)

Увеличить цикл i на 1

Увеличить цикл i на 1

А

Конец

Рисунок 2.1 – Алгоритм обучения нейронной сети

## **Алгоритм функционирования нейронной сети**

Начало

В

А

Ввод(x1;name)

p = p + 1;

p = p - 1;

k = k + 1;

k = 0;

p = 0;

Пока не будет достигнут конец файла будет продолжаться цикл i

Открыть файл с именем name

Закрыть файл name

work1(y1, x1, w, f1, F1)

work2(y2, x1, w, f2, F2)

work3(y3, x1, w, f3, F3)

work(Y, F1, F2, F3, Y1)

Цикл до момента окончания файла

Ввод( w[k][p])

Вывод("Ответ сети " , Y1)

Нет

Да

(p == 0)

Конец

А

В

Рисунок 2.2 – Алгоритм функционирования нейронной сети

## Алгоритм функционирования нейронной сети представляет из себя открытие заранее подготовленного файла с обученными весами, после чего пользователь вводит входное значение нейрона, нейронная сеть просчитывает результат и ему выдается ответ сети.

## **Вывод**

В данной главе были рассмотрены общий алгоритм работы программного средства, алгоритм функционирования нейронной сети, алгоритм обучения нейронной сети. Были проведены их описания и схемы работы. Все приведенные выше алгоритмы используются для дальнейшего проектирования модулей и лежат в основе для реализации основных функций.

# **3. Программное конструирование**

На основе выбранных алгоритмов решения поставленной задачи выполнено программное конструирование, предполагающее выбор соответствующих требованиям сред разработки и программирования и определение основных функций, составляющих будущее программное средство.

## **3.1 Выбор языка программирования**

Для реализации разработанного алгоритма был выбран язык программирования С++.

C++ (читается си-плюс-плюс) — компилируемый, статически типизированный язык программирования общего назначения, на котором можно создавать программы любого уровня сложности.  
Более 20 лет этот язык находится в тройке самых популярных и востребованных языков программирования. (В этом можно убедиться, посетив сайт TIOBE).

С++ – это мультипарадигмальный язык (от слова парадигма – стиль написания компьютерных программ), включающий широкий спектр различных стилей и технологий программирования. Часто его причисляют к объектно-ориентированным языкам, но, строго говоря, это не так. В процессе работы разработчик получает абсолютную свободу в выборе инструментов для того, чтобы задача, решаемая с помощью того или иного подхода, была решена максимально эффективно. Иными словами, С++ не понуждает программиста придерживаться только одного стиля разработки программы (например, объектно-ориентированного).[5]

## **3.2 Выбор среды программирования**

Для реализации разработанного алгоритма была выбрана среда программирования Visual Studio 2019.

Visual Studio – инструментальная среда разработки, включающая в себя интегрированную среду разработки, редактор исходного кода, встроенный отладчик. Многие другие инструменты возможно получить благодаря подключению плагинов – сторонних расширений. Был выбран по причине наличия интеграции с Unity 3d, поддержки языка программирования С#, наличие возможности написание собственных Unit-тестов. Также это наиболее изученная разработчиком платформа, что позволяет быстрее разрабатывать приложения не тратя время на изучения нового материала. [6]

## **3.3 Описание основных функций**

Разрабатываемое программное средство состоит из нескольких функций, которые задействуются в главной. Список основных функций предоставлен в таблице 1.

### **3.3.1 Используемые функции**

К основным функциям, используемым в программе относятся:

* wwf – функция, необходимая для реализации обучения из файла
* write –функция, необходимая для запись весов
* education – функция обучения
* work1 – функция, полного расчета 1 нейрона
* work2 – функция, полного расчета 2 нейрона
* work3 – функция, полного расчета 3 нейрона
* work – функция, для расчета итогового результата
* main – основная функция программы
* также для работы с файлами используются объекты класса ofstream и ifstream

Классы ofstream и ifsteam используют следующие методы:

1.open – открытие файла

2.close – закрытие файла

3.eof – поиск окончания файла

Подробно данные методы и функции описаны в таблице 1.

Таблица 3.1 – Описание функций и методов.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание метода | Возвращаемое значение |
| wwf | double p[5][3],double \*x1,double \*a,double \*t,int \*u | Функция используется для построчной записи из файла в массив входных элементов, необходимых для обучающей выборки сети | нет |
| write | double w[3][2],string name | Функция используется для записи весов в файл, веденный пользователем | нет |
| education | double\* a, double\* x1, double\* Y1,double \*d, double \*t, double \*Y, double \*d1, double \*d2, double \*d3, double \*d4, double \*d5, double \*d6, double \*f1, double \*f2, double \*f3, double w[3][2] | Функция используется для вычисления ошибки каждого из весов с последующим их изменением и выведением на экран исправленных значений | нет |
| Work1 | double \*y1, double \*x1, double w[3][2], double \*f1, double \*F1 | Функция, вычисляющая полный расчет 1 нейрона | нет |
| Work2 | double \*y2, double \*x1, double w[3][2], double \*f2, double \*F2 | Функция, вычисляющая полный расчет 2 нейрона | нет |
| Work3 | double \*y3, double \*x1, double w[3][2], double \*f3, double \*F3 | Функция, вычисляющая полный расчет 3 нейрона | нет |
| Work | double \*Y, double \*F1, double \*F2, double \*F3, double \*Y1 | Функция, вычисляющая конечный ответ сети | нет |
| main | нет | Основная функция программы, где прописаны все алгоритмы и циклы | 0 |
| open | name | Метод, при помощи которого происходит открытие файла введенного пользователем | нет |
| close | name | Метод, при помощи которого происходит закрытие файла введенного пользователем | нет |
| eof | нет |  | Возвращает значение 1, если файл пуст и 0, если файл не пуст |

Продолжение таблицы 3.1

### **3.3.2 Реализация обучения сети**

Ключевыми являются функции work, work1-3 именно в них происходит непосредственно работа программы, а также education, где происходит непосредственно вычисление ошибки и изменение весов. Функции work, work1-3 используются в начале программы для вычисления ошибки, которое реализовано методом обратного распространения ошибки в education.

Здесь изначально происходит выбор обучения: выборкой или по 1 ввод с клавиатуры. Далее после ввода названия файла куда будут записываться веса, задаются изначальные веса случайным образом и записываются в двумерный массив. Потом программа входит в цикл в котором непосредственно и происходит обучение сети и запись измененных весов в файл.

if (N2 == 2) {

cout << "ВВедите входное значение нейрона" << endl; cin >> x1;

cout << "Введите скорость обучения(от 0.1 до 0.9)" << endl; cin >> a;

cout << "Введите правильный ответ" << endl; cin >> t;

}

if (N2 == 1) {

wwf(p, &x1, &a, &t, &u);

}

cout << "Введите название обучающего файла для записи нейронной сети " << endl;

cin >> name;

if (O == 0) {

cout << "Значения Исходных Весов: " << endl;

cout << endl;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) {

w[i][j] = (rand() % 70) \* 0.01;

cout << "Число " << w[i][j] << " ";

}

}

}

cout << endl << endl;

for (int i = 0; i < 200; i++)

{

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

education(&a, &x1, &Y1, &d, &t, &Y, &d1, &d2, &d3, &d4, &d5, &d6, &f1, &f2, &f3, w);

write(w, name);

cout << "Ответ сети " << Y1 << endl << endl;

}

cout << "Сеть обучена на данном примере!" << endl << endl;

Ниже будет представлены реализации функций work, work1-3 и education.

void education( double\* a, double\* x1, double\* Y1,double \*d, double \*t, double \*Y, double \*d1, double \*d2, double \*d3, double \*d4, double \*d5, double \*d6, double \*f1, double \*f2, double \*f3, double w[3][2]){

(\*d) = (\*t) - (\*Y1);//вычисление ошибки

(\*d4) = (\*d) \* w[1][1];

(\*d6) = (\*d) \* w[2][1];

(\*d5) = (\*d) \* w[2][0];

(\*d1) = (\*d4) \* w[0][0];

(\*d2) = (\*d5) \* w[0][1];

(\*d3) = (\*d6) \* w[1][0];

w[0][0] = w[0][0] + (\*d1) \* (\*f1) \* (1 - (\*f1)) \* (\*x1) \* (\*a);

w[0][1] = w[0][1] + (\*d2) \* (\*f2) \* (1 - (\*f2)) \* (\*x1) \* (\*a);

w[1][0] = w[1][0] + (\*d3) \* (\*f3) \* (1 - (\*f3)) \* (\*x1) \* (\*a);

w[1][1] = w[1][1] + (\*d4) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

w[2][0] = w[2][0] + (\*d6) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

w[2][1] = w[2][1] + (\*d5) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

cout << "Значения Иcправленных Весов: " << endl;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) {

cout << "Число "<<i<<j<<" "<< w[i][j] << " " ;

cout << endl;

}

}

cout << endl;

}

void work( double \*Y, double \*F1, double \*F2, double \*F3, double \*Y1) {

(\*Y) = ((\*F1) + (\*F2) + (\*F3));

(\*Y1) = 1 / (1 + exp(-\*Y));

}

void work1( double \*y1, double \*x1, double w[3][2], double \*f1, double \*F1) {

(\*y1) = (\*x1) \* (w[0][0]);

(\*f1) = 1 / (1 + exp(-\*y1));//полный расчет 1 нейрона

(\*F1) = (\*f1) \* (w[0][1]);

}

void work2( double \*y2, double \*x1, double w[3][2], double \*f2, double \*F2) {

(\*y2) = (\*x1) \* (w[2][0]);

(\*f2) = 1 / (1 + exp(-\*y2));//полный расчет 2 нейрона

(\*F2) = (\*f2) \* (w[2][1]);

}

void work3( double \*y3, double \*x1, double w[3][2], double \*f3, double \*F3) {

(\*y3) = (\*x1) \* (w[1][0]);

(\*f3) = 1 / (1 + exp(-\*y3));//полный расчет 3 нейрона

(\*F3) = (\*f3) \* (w[1][1]);

}

### **3.3.3 Реализация работы сети**

Здесь после ввода пользователем входного значения и имени файла с обученными весами происходит открытие этого файла и запись в массив всех значений весов, после чего файл закрывается. И через функции work и work1-3 вычисляется ответ сети.

cout << "ВВедите входное значение " << endl;

cin >> x1;

cout << "ВВедите название файла с весами " << endl;

cin >> name;

ifstream fin;

fin.open(name);

int k=0;

int p = 0;

while (!fin.eof()) {

fin >> w[k][p];

if (p == 0) {

p = p + 1;

}

else {

p = p - 1;

k = k + 1;

}

}

fin.close();

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

cout << "Ответ сети "<< Y1 << endl<<endl;

## **3.4 Выводы по главе**

В данной главе был обоснован выбор языка программирования, а также

среда программирования для создаваемого программного средства. Кроме того, были описаны основные функции программного средства. В таблице было приведено описание каждой из функций с указанием типа входных параметров и возвращаемого значения.

# **Тестирование программного средства**

## **4.1 Контрольные примеры обучения и работы нейронной сети**

Для тестирования корректности работы разработанного программного

средства создается тестовая выборка контрольных примеров.

Первый пример выборки представляет собой входное значение, скорость обучения и правильный ответ сети для обучения.

Пример 1. Вычислить значение Sin(1)=x.

Решение. Sin(1)=0.841(округленное)

Кроме данного примера в тестовую выборку были добавлены еще 4 примера, такие как Sin(2), Sin(3), Sin(4), Sin(5).

## **4.2 Результаты тестирования программного средства**

Рассмотрим решение первого контрольного примера с помощью

разработанного программного средства.

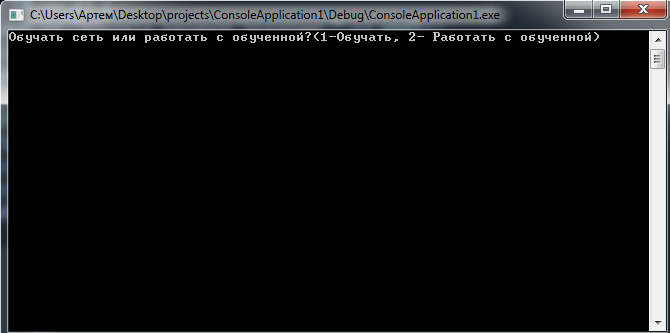
Старт консольного окна программы приведен на рисунке 4.1.

Рисунок 4.1 – Старт консольного окна

После того как пользователь сделал выбор обучения сети и далее выбор обучения сети по 1 примеру. Ему предоставляется возможность ввести входное значение, скорость обучения, правильный ответ и название файла, куда будут записываться измененные весы (4.2).

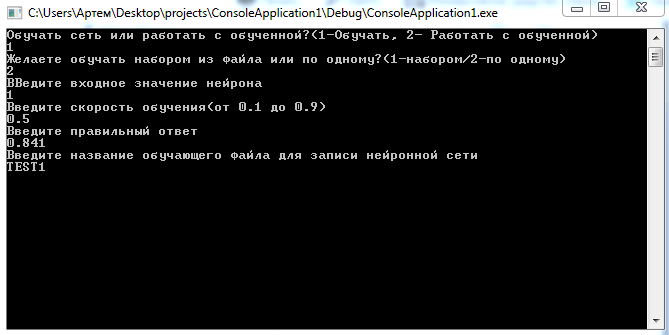


Рисунок 4.2 – Рисунок интерфейса работы системы

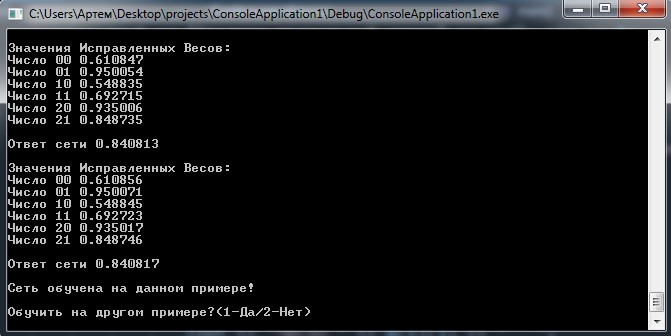
Далее после ввода всех вышеупомянутых значений происходит изменения весов и пользователю выдается ответ обучения сети. (4.3)

Рисунок 4.3 – Рисунок интерфейса работы системы

После обучения можно перейти в раздел работы сети, чтобы наглядно убедится в ее работоспособности (4.4).

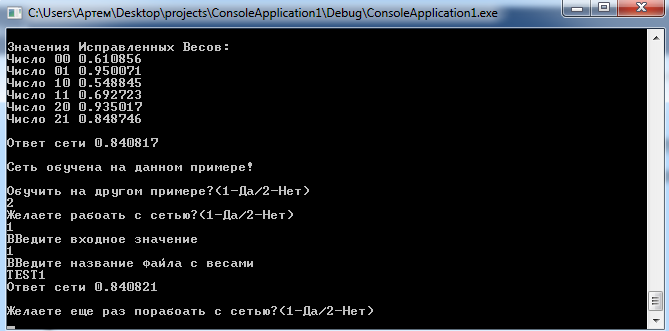


Рисунок 4.4 – Рисунок интерфейса работы системы

Результат решения программного средства совпадает с результатом при помощи математических вычислений.

## **4.3 Выводы по главе**

В данной главе была показана работа программного средства аппроксимация функций синуса и косинуса нейронной сетью. Были продемонстрированы все сцены разрабатываемого программного средства с кратким описанием их работы. Программное средство было протестировано более 10 раз, на основании чего, можно сделать вывод, что разработанное программное средство обладает всем требуемым функционалом и решает все поставленные задачи.

# **ЗАКЛЮЧЕНИЕ**

В рамках данной работы была создана программное средство с консольным интерфейсом, для проведения аппроксимации функций синуса и косинуса нейронной сетью.

В ходе выполнения работы был построен аналитически обоснованный алгоритм решения поставленной задачи и разработано соответствующее программное средство, конечный исходный код которого можно просмотреть в Приложении Б. Во время написания программного средства была выполнены большая часть поставленных задач и на их основе, была выполнена проверка корректной работоспособности программного средства.

Реализованы алгоритмы:

* + - * Алгоритм функционирования нейронной сети
      * Алгоритм работы нейронной сети

Получены и закреплены навыки разработки программного средства на языке программирования С++ в среде Microsoft Visual Studio.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1**.** Крючин, О.В. Искусственные нейронные сети и кластерные системы. Реализация нейронного симулятора: [Вестник российских университетов. Математика](https://cyberleninka.ru/journal/n/vestnik-rossiyskih-universitetov-matematika), 2010 – 306с.

2. Голубинский, А.Н. Методы аппроксимации экспериментальных данных и построения моделей: [Вестник Воронежского института МВД России](https://cyberleninka.ru/journal/n/vestnik-voronezhskogo-instituta-mvd-rossii), 2007 – 1 с.;

3. Шестаков С. Аппроксимация функции. Основные понятия и определения, 2017(06.04.20) <https://docplayer.ru/34794613-Approksimaciya-funkciy-osnovnye-ponyatiya-i-opredeleniya.html>;

4. А. С. Шведов, “Аппроксимация функций с помощью нейронных сетей и нечетких систем”, *Пробл. управл.*, 2018, № 1, 21–29;

5. Общие сведения о языке программирования С++ // Образовательный онлайн-сервис inf.w [Электронный ресурс], редактировано 22-08-2019. URL.: <http://inf-w.ru/?page_id=8655> (дата обращения: 04.06.2020);

6. Microsoft Visual Studio // Образовательный онлайн-сервис WIKIPEDIA [Электронный ресурс], редактировано 04.06.2020. URL.: <https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio#Visual_Studio_2019> (дата обращения 04.06.2020)

# **ПРИЛОЖЕНИЕ А**

# **Техническое задание**

СОГЛАСОВАНО

Доцент каф «ПОВТиАС»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Скляренко А.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

**А.1 Введение**

**А.1.1 Наименование программного средства**

Наименование программного средства – «Аппроксимация тригонометрических функций косинуса и синуса нейронной сетью».

**А.1.2 Область применения**

Программное средство может применяться в учебных заведениях, а также в научных заведениях, которым необходимо вычисление синуса или косинуса.

**А.2 Основание для разработки**

Разработка ведется на основании документа «Учебный план для студентов ВУЗа» направления 09.03.04 «Программная инженерия» кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» факультета «Информатика и вычислительная техника» Донского Государственного Технического Университета.

**А.3 Назначение разработки**

**А.3.1 Функциональное назначение**

Функциональное назначение программного средства заключается в обучении нейронной сети для последующего вычисления косинуса и синуса нейронной сетью.

**А.3.2 Эксплуатационное назначение**

Эксплуатационное назначение состоит в использовании программного средства на ноутбуках/ПК с операционной системой Linux Ubuntu 18.04, Windows 10.

**А.4 Требования к программе**

**А.4.1 Требования к функциональным характеристикам**

Программное средство должно осуществлять следующие функции:

* Обучение сети
* Работа сети
* Запись весов в файл
* Возможность обучать сеть по 1 примеру или же обучающей выборкой из файла
* Возможность для пользователя вводить название для файла с весами

**А.4.2 Требования к надежности**

Специальных требований к надежности не предъявляется

**А.4.2.1 Входные данные**

В качестве входных данных пользователь вводит входное значение синуса или косинуса, скорость обучения нейронной сети ответ ( в случае обучения сети), правильный ответ ( в случае обучения сети), а также в случае перемещения по консольному меню для выбора определенного действия цифры (1-Да/2-Нет).

**А.4.2.2 Выходных данные**

Выходные данные представляются в виде выведенных в консоль значений весов, а также ответа сети (в случае обучения) ,или же только ответа сети, т.е. полученного результата вычисления синуса или косинуса (в случае работы с сетью).

**А.4.3 Условия эксплуатации**

Для стабильного функционирования и оптимальной работы программного продукта необходимо соблюдение всех требований и правил эксплуатации компьютерной техники. Каких-либо требований к пользователю данного приложения нет.

**А.4.4 Требования к составу и параметрам технических средств**

В состав технических средств входят ноутбуки/ПК с операционной системой Linux Ubuntu 18.04, Windows 10:

* процессор Intel Pentium III, AMD, Intel Celeron с тактовой частотой от 75 mhz и выше;
* оперативная память от 128 Мб;
* жесткий диск c объемом памяти не менее 60 Мб свободного дискового пространства;
* видеокарта SVGA, с объемом оперативной памяти 512 Мб;
* монитор с размером экрана 15 дюймов, поддерживающий разрешение 800\*600 при 256 цветах;
* клавиатура стандартная PS/2 (101/102 клавиши).

Дополнительных требований и ограничений к составу и параметрам технических средств не вводится.

**А.4.5 Требования к информационной и программной совместимости**

Для функционирования программного средства необходима операционная система Linux Ubuntu 18.04, Windows 10. Исходные коды должны быть реализованы на языке С++.

**А.4.6 Требования к упаковке и маркировки**

К упаковке и маркировки специальных требований не предъявляются.

**А.4.7 Требования к транспортировке и хранению**

К транспортировке и хранению специальных требований не предъявляется.

**А.4.8 Оценочные требования к функционалу программного средства**

Таблица 1 - Оценочные требования к функционалу программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Реализуемые программным средством функции | Степень самостоятельности разработки («новое», «модиф.», «заимств.») | |
| Алгоритм | ПС |
| 1 | Обучение сети | Заимств. | Новое |
| 2 | Работа сети | Заимств. | Новое |
| 3 | Запись весов в файл | Заимств. | Новое |
| 4 | Возможность обучать сеть по 1 примеру или же обучающей выборкой из файла | Заимств. | Новое |
| 5 | Возможность для пользователя вводить название для файла с весами | Заимств. | Новое |

Окончание таблицы 1

**А.5 Требования к программной документации**

Программная документация должна состоять из следующего:

* титульный лист;
* лист задания на учебную ознакомительную практику;
* пояснительная записка к учебной ознакомительной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код приложения по ГОСТ 19.401-79 ЕСПД.

**А.6 Стадии и этапы разработки**

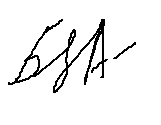
Реализация программного средства состояла из следующих этапов:

* проведение аналитического обзора области линейных уравнений и постановка задачи (11.02.2020 – 5.04.2020);
* алгоритмическое конструирование (6.04.2020 – 19.04.2020);
* программная реализация (20.04.2020 – 17.05.2020);
* тестирование приложения (18.05.2020 – 20.05.2020);
* разработка отчета (21.05.2020 – 5.06.2020).

**А.7 Порядок контроля и приемки**

Порядок и контроль приемки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие корректно работающего программного средства и отчета, предоставленного в печатном виде.

Разработчик технического задания /Бережнов Артем Сергеевич/

«\_\_5\_» \_\_\_апреля\_\_\_\_\_2020 г. \_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

# **ПРИЛОЖЕНИЕ Б**

**Исходный код программного средства**

Полный код программного средства находится на съемном носителе, прилагаемом к данной работе.

Листинг Б.1 – Исходный код программного средства

#include <iostream>

#include <math.h>

#include <cstdlib>

#include <fstream>

using namespace std;

void wwf(double p[5][3],double \*x1,double \*a,double \*t,int \*u) {

ifstream fin;

fin.open("NS.txt");

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 3; j++)

{

fin >> p[i][j];

}

}

\*x1 = p[\*u][0];

\*a = p[\*u][1];

\*t = p[\*u][2];

\*u += 1;

fin.close();

}

void write( double w[3][2],string name)

{

ofstream fout;

fout.open(name);

for (int j = 0; j < 3; j++) {

for (int i = 0; i < 2; i++) {

fout << w[j][i] << endl;

}

}

fout.close();

}

void education( double\* a, double\* x1, double\* Y1,double \*d, double \*t, double \*Y, double \*d1, double \*d2, double \*d3, double \*d4, double \*d5, double \*d6, double \*f1, double \*f2, double \*f3, double w[3][2]){

(\*d) = (\*t) - (\*Y1);//вычисление ошибки

(\*d4) = (\*d) \* w[1][1];

(\*d6) = (\*d) \* w[2][1];

(\*d5) = (\*d) \* w[2][0];

(\*d1) = (\*d4) \* w[0][0];

(\*d2) = (\*d5) \* w[0][1];

(\*d3) = (\*d6) \* w[1][0];

w[0][0] = w[0][0] + (\*d1) \* (\*f1) \* (1 - (\*f1)) \* (\*x1) \* (\*a);

w[0][1] = w[0][1] + (\*d2) \* (\*f2) \* (1 - (\*f2)) \* (\*x1) \* (\*a);

w[1][0] = w[1][0] + (\*d3) \* (\*f3) \* (1 - (\*f3)) \* (\*x1) \* (\*a);

w[1][1] = w[1][1] + (\*d4) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

w[2][0] = w[2][0] + (\*d6) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

w[2][1] = w[2][1] + (\*d5) \* (\*Y1) \* (1 - (\*Y1)) \* (\*x1) \* (\*a);

cout << "Значения Иcправленных Весов: " << endl;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) {

cout << "Число "<<i<<j<<" "<< w[i][j] << " " ;

cout << endl;

}

}

cout << endl;

}

void work( double \*Y, double \*F1, double \*F2, double \*F3, double \*Y1) {

(\*Y) = ((\*F1) + (\*F2) + (\*F3));

(\*Y1) = 1 / (1 + exp(-\*Y));

}

void work1( double \*y1, double \*x1, double w[3][2], double \*f1, double \*F1) {

(\*y1) = (\*x1) \* (w[0][0]);

(\*f1) = 1 / (1 + exp(-\*y1));//полный расчет 1 нейрона

(\*F1) = (\*f1) \* (w[0][1]);

}

void work2( double \*y2, double \*x1, double w[3][2], double \*f2, double \*F2) {

(\*y2) = (\*x1) \* (w[2][0]);

(\*f2) = 1 / (1 + exp(-\*y2));//полный расчет 2 нейрона

(\*F2) = (\*f2) \* (w[2][1]);

}

void work3( double \*y3, double \*x1, double w[3][2], double \*f3, double \*F3) {

(\*y3) = (\*x1) \* (w[1][0]);

(\*f3) = 1 / (1 + exp(-\*y3));//полный расчет 3 нейрона

(\*F3) = (\*f3) \* (w[1][1]);

}

int main() {

setlocale(LC\_ALL, "ru");

double p[5][3];//массив для обучающей выборки

const int ROWS = 3;//строка массива

const int COL = 2;//столбец массива

int R; int N = 1; int N1 = 1; int O = 0; int N2 = 0; int u = 0;//переменные для циклов

string name;//ввод-вывод

double x1, w[ROWS][COL], y1, y2, y3, f1, f2, f3, F1, F2, F3, Y, Y1, t, d, d1, d2, d3, d4, d5, d6, a;//переменные для обучения

cout << "Обучать сеть или работать с обученной?(1-Обучать, 2- Работать с обученной)" << endl;

cin >> R;

if (R == 1) {

cout << "Желаете обучать набором из файла или по одному?(1-набором/2-по одному)" << endl;

cin >> N2;

}

while (R == 1) {

if (N2 == 2) {

cout << "ВВедите входное значение нейрона" << endl; cin >> x1;

cout << "Введите скорость обучения(от 0.1 до 0.9)" << endl; cin >> a;

cout << "Введите правильный ответ" << endl; cin >> t;

}

if (N2 == 1) {

wwf(p, &x1, &a, &t, &u);

}

cout << "Введите название обучающего файла для записи нейронной сети " << endl;

cin >> name;

if (O == 0) {

cout << "Значения Исходных Весов: " << endl;

cout << endl;

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 2; j++) {

w[i][j] = (rand() % 70) \* 0.01;

cout << "Число " << w[i][j] << " ";

}

}

}

cout << endl << endl;

for (int i = 0; i < 200; i++)

{

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

education(&a, &x1, &Y1, &d, &t, &Y, &d1, &d2, &d3, &d4, &d5, &d6, &f1, &f2, &f3, w);

write(w, name);

cout << "Ответ сети " << Y1 << endl << endl;

}

cout << "Сеть обучена на данном примере!" << endl << endl;

cout << "Обучить на другом примере?(1-Да/2-Нет)" << endl;

cin >> N;

if (N == 1) {

O = 1;

}

while (N == 1) {

if (N2 == 2) {

cout << "ВВедите входное значение нейрона" << endl; cin >> x1;

cout << "Введите скорость обучения(от 0.1 до 0.9)" << endl; cin >> a;

cout << "Введите правильный ответ" << endl; cin >> t;

}

if (N2 == 1) {

wwf(p, &x1, &a, &t, &u);

}

for (int i = 0; i < 200; i++)

{

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

education(&a, &x1, &Y1, &d, &t, &Y, &d1, &d2, &d3, &d4, &d5, &d6, &f1, &f2, &f3, w);

write(w,name);

cout << "Ответ сети " << Y1 << endl << endl;

}

cout << "Сеть обучена на данном примере!" << endl << endl;

cout << "Обучить на другом примере?(1-Да/2-Нет)" << endl;

cin >> N;

}

cout << "Желаете рабоать с сетью?(1-Да/2-Нет)" << endl;

cin >> R;

if (R == 1) {

R = 2;

}

else {

exit(1);

}

}

while (R == 2) {

cout << "ВВедите входное значение " << endl;

cin >> x1;

cout << "ВВедите название файла с весами " << endl;

cin >> name;

ifstream fin;

fin.open(name);

int k=0;

int p = 0;

while (!fin.eof()) {

fin >> w[k][p];

if (p == 0) {

p = p + 1;

}

else {

p = p - 1;

k = k + 1;

}

fin.close();

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

cout << "Ответ сети "<< Y1 << endl<<endl;

cout << "Желаете еще раз порабоать с сетью?(1-Да/2-Нет)" << endl;

cin >> R;

if (R == 1) {

R = 2;

}

else{

cout << "Желаете обучить сеть?(1-Да/2-Нет)" << endl;

cin >> N;

while (N==1) {

cout << "ВВедите входное значение нейрона" << endl; cin >> x1;

cout << "Введите скорость обучения(от 0.1 до 0.9)" << endl; cin >> a;

cout << "Введите правильный ответ" << endl; cin >> t;

for (int i = 0; i < 200; i++)

{

work1(&y1, &x1, w, &f1, &F1);

work2(&y2, &x1, w, &f2, &F2);

work3(&y3, &x1, w, &f3, &F3);

work(&Y, &F1, &F2, &F3, &Y1);

education(&a, &x1, &Y1, &d, &t, &Y, &d1, &d2, &d3, &d4, &d5, &d6, &f1, &f2, &f3, w);

write(w,name);

cout << "Ответ сети " << Y1 << endl << endl;

}

cout << "Сеть обучена на данном примере!" << endl << endl;

cout << "Желаете обучить сеть еще раз?(1-Да/2-Нет)" << endl;

cin >> N1;

if(N1==2){

cout << "Желаете работать с сетью?(1-Да/2-Нет)" << endl;

cin >> R;

if (R == 1) {

R = 2;

}

else {

exit(1);

}

}

}

if (N == 2) {

exit(1);

}

}

}

return 0;

}

}