Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

Разраб.

Дорохова К.С.

Руководит

Атаян А.М.

Многослойный персептрон

Лист.

Листов

57

ДГТУ

Кафедра «ПОВТиАС»

[Введение 8](#_Toc168696343)

[1 Аналитический обзор предметной области 10](#_Toc168696344)

[1.1 Обзор предметной области 10](#_Toc168696345)

[1.2 Обзор модели «Многослойный персептрон» 11](#_Toc168696346)

[1.3 Постановка задачи 14](#_Toc168696347)

[1.4 Выводы по главе 15](#_Toc168696348)

[2 Алгоритмическое конструирование 16](#_Toc168696349)

[2.1 Общий алгоритм работы 16](#_Toc168696350)

[2.2 Алгоритм обучения 17](#_Toc168696351)

[2.3 Алгоритм прямого распространения 18](#_Toc168696351)

[2.4 Алгоритм обратного распространения 19](#_Toc168696351)

[2.5 Выводы по главе 20](#_Toc168696352)

[3 Программное конструирование 21](#_Toc168696353)

[3.1 Обоснование выбора средств разработки 21](#_Toc168696354)

[3.2 Описание программной реализации 22](#_Toc168696355)

[3.4 Выводы по главе 25](#_Toc168696357)

[4 Демонстрация работы программного средства 26](#_Toc168696358)

[4.1 Описание контрольного примера 26](#_Toc168696359)

[4.2 Описание процесса работы с программой 26](#_Toc168696360)

[4.3 Выводы по главе 29](#_Toc168696361)

[Заключение 30](#_Toc168696362)

[Перечень использованных информационных ресурсов 31](#_Toc168696363)

[Приложение А Листинг программы 33](#_Toc168696364)

Введение

Современный мир невозможно представить без искусственных нейронных сетей, ведь они оказывают огромное влияние практически на все сферы деятельности человека. Нейросеть — это программа, которая умеет обучаться на основе данных и примеров, то есть работает по принципу нервной системы живого организма. Нейросети позволяют автоматизировать сложные процессы, анализировать огромные объемы данных, принимать обоснованные решения с высокой точностью и скоростью, а также экономят время и человеческие ресурсы. В основе всех этих достижений лежит многослойный персептрон – класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоев нейронов: входного, скрытого и выходного [1]. Многослойный персептрон остается одной из наиболее популярных и широко используемых архитектур нейронных сетей благодаря своей универсальности и гибкости. В эпоху больших данных и развития искусственного интеллекта многослойный персептрон находит применение во многих областях, а также широко используется для решения множества задач машинного обучения: распознавание образов, обработка, сегментация или генерирование изображений, классификация, обработка естественного языка, анализ данных в научных исследованиях, прогнозирование, автоматизация и оптимизация процессов, решение задач с быстрым и точным результатом.

Искусственные нейронные сети оказывают огромное влияние на все сферы деятельности, создавая более удобные, доступные и быстрые способы решения задач. Применение многослойного персептрона позволяет эффективно и быстро решать задачи, автоматизируя сложные процессы, что значительно упрощает работу и дает толчок к быстрому развитию науки и различных областей за счет сокращения времени.

Целью данной работы является упрощение и ускорение решения некоторых задач, а также получение быстрого и точного результата на основе работы многослойного персептрона.

В первой главе на основании сравнения существующих архитектур искусственных нейронных сетей, приведен обзор предметной области поставленной задачи, а также сформулировано ее условие.

Во второй главе представлены алгоритмы, используемые для решения поставленной задачи.

В третьей главе описано обоснование выбора языка программирования и представлена структура базы данных, которая была использована для создания программного средства.

В четвертой главе представлена работа созданного программного средства.

1 Аналитический обзор предметной области

В данном разделе будут проанализированы существующие структуры нейронных сетей, приведены их особенности, а также будут рассмотрены их достоинства и недостатки. Здесь же будут сформулированы конечные цели и задачи для выполняемой работы.

* 1. Обзор предметной области

В современном мире многие люди стремятся к эффективному и быстрому решению сложных задач в области анализа данных, распознавания образов и других задачах, для которых требуется машинное обучение. Данная потребность обусловлена необходимостью обработки огромных объемов информации и достижением высокой точности в условиях быстрого технологического прогресса и ограниченного количества времени.

В результате чего все чаще стали применяться такие мощные инструменты, как искусственные нейронные сети, которые помогают оптимизировать процессы, улучшить качество результатов и ускорить процесс решения задач.

Нейросети состоят из искусственных нейронов, которые соединяются между собой. Нейроны могут быть по-разному соединены друг с другом. Различаются и способы передачи данных, и формулы, которые их описывают [2]. Обычно нейросети делят на архитектуры: персептроны, многослойные персептроны, рекуррентные нейронные сети, сверточные нейронные сети, глубокие нейронные сети и многие другие. Сегодняшний мир нейронных сетей предлагает удивительное разнообразие инструментов для решения самых разнообразных задач. Выбор конкретного типа сети зависит от природы данных и целей, которые вы хотите достичь [3]. Однако одной из самых популярных моделей нейросетей является многослойный персептрон.

Многослойный персептрон является фундаментальной моделью в области машинного обучения благодаря своей универсальности и способности обрабатывать сложные задачи. Часто многослойный персептрон является либо самостоятельной нейросетью, либо блоком каком-нибудь мощной нейронной сети. Многослойный персептрон не требует сложной настройки и может быть запущен на любом устройстве, что делает его очень удобным в использовании. Эта универсальность делает многослойный персептрон мощным инструментом для решения широкого спектра задач машинного обучения.

* 1. Обзор модели «Многослойный персептрон»

Среди архитектур нейронных сетей существует множество предлагаемых моделей. Далее будут описаны такие модели, как: однослойный персептрон и многослойный персептрон.

Особенность однослойного персептрона состоит в том, что он состоит из одного слоя выходных узлов, ходы которой идут непосредственно на выходы через серию весов. Сумма произведений весов и входов рассчитывается в каждом узле, и если значение превышает некоторый порог (обычно 0), нейрон срабатывает и принимает активированное значение (обычно 1); в противном случае он принимает деактивированное значение (обычно −1). Нейроны с такой функцией активации также называют искусственными нейронами или линейными пороговыми юнитами. Однослойные персептроны способны изучать только линейно разделимые структуры, например такой персептрон не может выучить исключающее ИЛИ [9].

Многослойный персептрон состоит как минимум из трех слоев: входного, скрытого и выходного. За исключением входных, все нейроны использует нелинейную функцию активации. Количество скрытых слоев не ограничено. Многослойный персептрон способен обучаться нелинейным функциям благодаря нелинейным активационным функциям в нейронах скрытых слоев. В качестве активационных функций нейронов используются сигмоидальные: логистическая или гиперболический тангенс, а также могут использоваться пороговые функции и множество других в зависимости от задания. Входные сигналы не бинарные, а кодируются десятичными числами, нормированными к интервалу [0,1]. Многослойный персептрон является универсальным аппроксиматором, то есть теоретически способен аппроксимировать любую непрерывную функцию с желаемой точностью при наличии достаточного количества нейронов в скрытых слоях. Многослойные персептроны успешно применяются в различных областях, таких как распознавание образов, классификация, прогнозирование, обработка сигналов и многих других [9].

При обучении многослойного персептрона используется алгоритмы прямого распространения и обратного распространения ошибки, а также обучение с учителем [10].

Обучение с учителем: дается набор отмеченных данных, для которых мы уже знаем какой именно является нашим правильным выходом, и он должен быть аналогичен набору, имея представление о том, что существует связь между входом и выходом. Задачи обучения с учителем подразделяются на задачи «регрессии» и «классификации». В задачах регрессии мы пытаемся предсказать результаты на непрерывном выходе, что означает, что мы пытаемся сопоставить входные переменные с некоторой непрерывной функцией. В задачах классификации мы стараемся предсказать результаты на дискретном выходе. Другими словами, мы пытаемся сопоставить входные переменные по разным категориям [10].

На этапе прямого распространения входные данные проходят через сеть от входного слоя к выходному. Каждый нейрон в скрытых и выходном слоях вычисляет взвешенную сумму своих входов и применяет к ней функцию активации. Формула для вычисления выхода нейрона [10]:

(1.1)

где: - выход нейрона; - функция активации; - вес i-го входа нейрона; - i-й вход нейрона; – смещение (bias) нейрона; – количество нейронов прошлого слоя;

После прямого прохода вычисляется ошибка на выходном слое (разница между предсказанным и истинным значением). Затем эта ошибка распространяется обратно через сеть, чтобы вычислить ошибки для весов и смещений. Цель обратного распространения - минимизировать ошибку, корректируя веса и смещения в направлении, противоположном ошибке.

Формула вычисления ошибки нейрона:

(1.2)

Где:

* – i-й выход нейрона;
* - предсказанное значение;
* - ошибка нейрона;
* – производная функции активации, которая принимает значение i-ого выхода нейрона;

Формула обновления весов:

Где:

* – i-й вес между первым и вторым слоем;
* – i-й выход нейрона первого слоя;
* – i-ая ошибка нейрона второго слоя;
* –скорость обучения функции;

Формула обновления смещений:

Где:

* – i-й смещение;
* – i-ая ошибка нейрона следующего слоя;
* – скорость обучения функции;
  1. Постановка задачи

Целью работы является повышение эффективности и ускорение процессов различных задач с помощью работы и обучения многослойного персептрона.

Разрабатываемый многослойный персептрон должен содержать следующий основной функционал:

* реализация выбора вида обучения;
* реализация выбора задачи для обучения;
* реализация настройки количества нейронов, эпох и скорости обучения;
* реализация отображения результатов тестирования после обучения;
* реализация сохранения весов и смещений;
* реализация механизма заполнения случайными числами весов и смещений;
* реализация выбора начальных весов и смещений;
* реализация ввода заданий для обученной нейросети.

Для достижения поставленной цели необходимо решить следующие задачи:

* разработать необходимые алгоритмы;
* выбрать языки программирования;
* разработать интерфейс взаимодействия с пользователем;
* разработать прогрессивное веб-приложение;
* протестировать многослойный персептрон.
  1. Выводы по главе

В данном разделе были рассмотрены основные концепции и цели, которые реализуются в программе многослойного персептрона, основной задачей которого является оптимизация и автоматизация задач для их быстрого решения. Были проанализированы такие модели нейронной сети, как: персептрон и многослойный персептрон. На основании проведенного обзора для достижения выявленной цели за модель нейронной сети был взят многослойный персептрон, так как по сравнению с простым персептроном, многослойный персептрон обладает большей гибкостью и мощностью благодаря наличию скрытых слоев, что позволяет ему аппроксимировать более сложные нелинейные функции.

Таким образом, была поставлена задача по реализации программного средства, которое будет являться моделью многослойного персептрона, будущие функции которого были подробно описаны и объяснены.

2 Алгоритмическое конструирование

В данном разделе рассматриваются основные алгоритмы работы многослойного персептрона: общий алгоритм работы приложения, алгоритм обучения, алгоритм прямого распространения, алгоритм обратного распространения ошибки.

2.1 Общий алгоритм работы

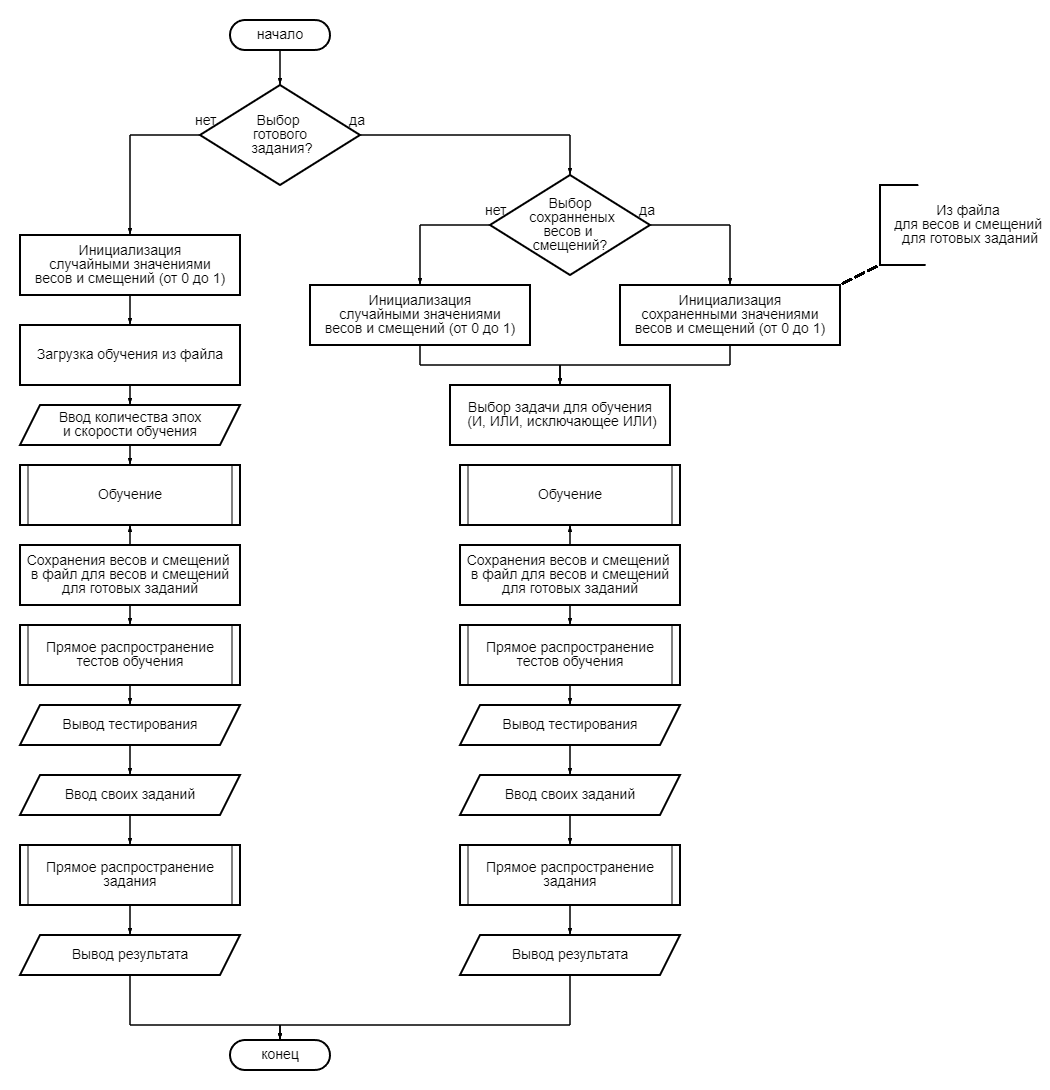


Рисунок 2.1 – Общий алгоритм работы многослойного персептрона

Общий алгоритм работы многослойного персептрона состоит из следующей последовательности действий:

* запуск многослойного персептрона;
* выбор готового или пользовательского задания;
* выбор сохраненных или случайных весов и смещений;
* инициализация весов и смещений;
* загрузка задания для обучения;
* обучение;
* сохранение новых весов и смещений;
* прямое распространение тестов обучения и вывод результата тестирования;
* ввод своих заданий для обученной нейросети и вывод результатов;
* выход из программы;

Общий алгоритм работы веб-приложения представлен на рисунке 2.1.

**2.2 Алгоритм обучения**

Алгоритм обучения выполняется после выбора режима обучения и задания для обучения.

Алгоритм обучения показан на рисунке 2.2.

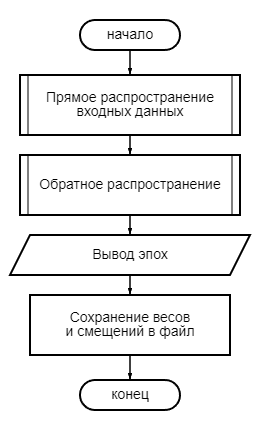


Рисунок 2.2 – Алгоритм обучения

**2.3 Алгоритм прямого распространения**

Алгоритм прямого распространения выполняется при вычислении результата после подачи входных данных в нейроны входного слоя. При этом данными заполняются все последующие нейроны каждого слоя, пока не будут вычислены значения нейронов выходного слоя, что и будет являться результатом работы многослойного персептрона.

Алгоритм прямого распространения показан на рисунке 2.3.

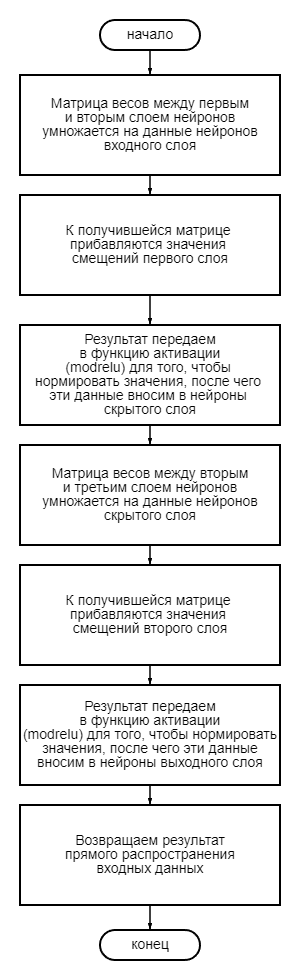


Рисунок 2.3 – Алгоритм прямого распространения

**2.4 Алгоритм обратного распространения ошибки**

Алгоритм обратного распространения ошибки выполняется после прямого распространения при обучении нейросети для обновления весов и смещений. При этом, начиная с выходного слоя, вычисляется ошибка для каждого нейрона.

Алгоритм обратного распространения показан на рисунке 2.4.

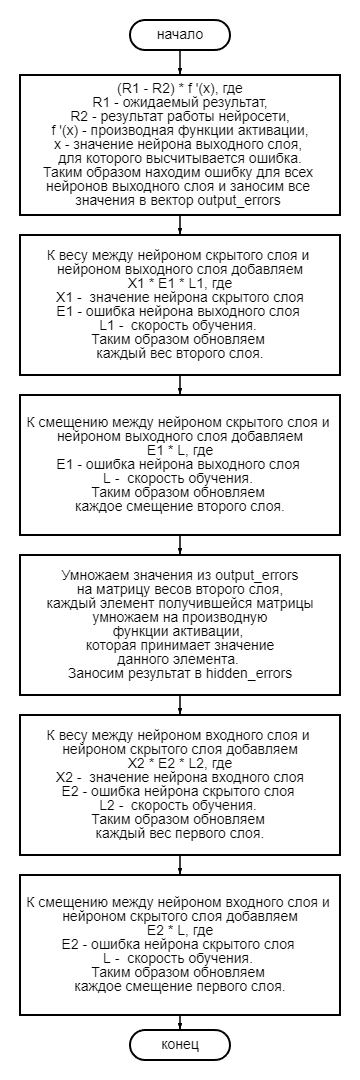


Рисунок 2.4 – Алгоритм обратного распространения

**2.6 Выводы по главе**

В данной главе было произведено алгоритмическое конструирование, то есть разработаны методы и процессы, основанные на алгоритмах и структурах данных, с целью достижения определенных результатов или решения задач. Этот подход позволяет систематизировать знания и опыт, а также обеспечивает эффективное решение сложных задач.

**3 Программное конструирование**

В данном разделе будут обоснованы выбор языка программирования и технологий, используемых для реализации программного средства и основания выбора среды программирования. Будут определены и описаны основные классы разрабатываемого приложения, структура проекта и его функции.

**3.1 Обоснование выбора средств разработки**

Для реализации программного средства был выбран текстовый редактор Visual Studio 2022.

Visual Studio - этот текстовый редактор, созданный Microsoft для операционных систем Windows, Linux и macOS. Это комплексная интегрированная среда разработки (IDE), которую можно использовать для записи, редактирования, отладки и сборки кода. Интегрированная среда разработки Visual Studio предоставляет множество функций, упрощающих написание кода. Например, исправление кода с помощью искусственного интеллекта. К этим средствам относятся GitHub Copilot и IntelliCode. Также он предоставляет инструменты для работы с Git, подсветку синтаксиса, возможности для рефакторинга кода, поддерживает множество языков и версий. Редактор обеспечивает широкие возможности настройки, включая пользовательские темы, настройки сочетания клавиш и файлы конфигурации. Более того, Visual Studio доступен бесплатно на официальном сайте Microsoft. [11]

Используемые библиотеки:

* iostream - заголовочный файл с классами, функциями и переменными для организации ввода-вывода [4];
* random - определяет средства для генерации случайных чисел с равномерным распределением [5];
* windows.h - это заголовочный файл, специфичный для Windows, который содержит объявления для всех функций в Windows API, все распространенные макросы для Windows, и все типы данных, используемые различными функциями и подсистемами [6];
* fstream - заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл [7];
* sstream - заголовочный файл с классами, функциями и переменными для организации работы со строками, через интерфейс потоков, в языке программирования C++ [8];

3.2 Описание программной реализации

Программа реализована на основе функциональной (или объектно-ориентированной) парадигмы программирования

Функции программы связаны с чтением входных данных и ожидаемых результатов из файлов, также осуществляются проверки и записи в файл.

В таблице 3.1 представлены реализованные функции.

Таблица 3.1 – Функции

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Входные параметры | Описание | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| clean | - | В случае ошибки несоответствия типов при вводе пользователем значений, очищается поток и выдается сообщение об ошибке | Текст о том, что произошла ошибка |
| read\_inputs | Входные и выходные данные логических задач (и, или, исключающее или) | Читает и записывает из файла входные данные для обучения | - |

Окончание таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Входные параметры | Описание | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| read\_targets | Входные и выходные (и, или, исключающее или) | Читает и записывает из файла выходные данные для обучения | - |
| read\_test | Ссылка на файл с входными и выходными данными для обучения, ссылки на векторы входных и выходных данных | Читает и записывает из файла входные и выходные данные для обучения собственному заданию | - |
| tasks | Выбор сохраненных или случайных весов и смещений, выбор задания, входные и выходные данные логических задач, ссылка на объект класса многослойный персептрон, выбор логического или своего задания | Данная функция выполняется, когда пользователь выбирает логическое задание для обучения. Вызывает функцию инициализации, обучения, выводит результат тестирования и позволяет вводить свои данные для обученной нейросети | - |
| check | Строка | Проверяет, состоит ли строка только из положительных чисел | true, если строка состоит только из положительных чисел, false в обратном случае |
| task\_test | Файл для сохранения весов и смещений, входные и выходные данные, объект класса многослойный персептрон, выбор логического или своего задания | Данная функция выполняется, когда пользователь выбирает собственное задание для обучения. Вызывает функцию инициализации, обучения, выводит результат тестирования и позволяет вводить свои данные для обученной нейросети | - |

Класс MultilayerPerceptron отвечает за моделирование многослойного

персептрона, состоящего из трех слоев нейронов.

В таблице 3.2 представлены методы класса MultilayerPerceptron.

Таблица 3.2 – Методы класса MultilayerPerceptron

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Входные параметры | Описание | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| Init | Выбор сохраненных или случайных чисел | Инициализирует веса и смещения значениями в отрезке [0; 1] | - |
| forward\_propagation | Входные и выходные данные | Вычисляет результат | Вектор с выходными данными |
| backpropagation\_updating\_weights | Тестовые входные и выходные данные, выходные данные после прямого распространения, скорость обучения | Обновление весов и смещений | - |
| matrix\_multiply | Две матрицы, количество входных и выходных нейронов, входные данные | Умножение двух матриц, получившаяся матрица умножается на производную функцию активации | Вектор с результатом вычислений |
| train | Входные и выходные данные, количество эпох, скорость обучения, выбор задания | Обучение нейросети выбранному заданию | - |
| file | Выбранное задание | Сохранение весов и смещений в файл | - |
| modrelu | Значение нейрона | Считает значение функции активации | - |
| modrelu\_derivative | Значение нейрона | Считает значение производной функции активации | - |
| matrix\_multiply\_sum | Две матрицы, количество входных и выходных нейронов, смещения | Умножение двух матриц, получившаяся матрица складывается со значениями смещений | Вектор с результатом вычислений |

3.3 Выводы по главе

В данной главе был обоснован выбор языка и среды программирования для программного средства, были выделены преимущества их выбора для данного приложения.

Были описаны основные функции программного средства, в описание которых входило: описание функций и методов, а также входные параметры, типы параметров и результата функции.

4 Демонстрация работы программного средства

В данном разделе продемонстрирована работа программного средства на основе скриншотов.

4.1 Описание контрольного примера

Для демонстрации работы программного средства рассматривается решение одной логической задачи - исключающее ИЛИ, а также одной пользовательской задачи, которая сортирует входные данные в порядке убывания, с входными данными: 0 1 0.

4.2 Описание процесса работы с программой

После запуска программы на экране появляется выбор задания для обучения, представленный на рисунке 4.1. Если пользователь выбирает готовое задание, он должен выбрать, какие веса и смещения использовать – сохраненные с прошлых обучений или же случайные, после чего он должен выбрать логическую задачу для обучения, что показано на рисунке 4.2. После 1000 эпох программа выводит результат тестирования, после чего пользователь может вводить свои задания или выйти из данного блока и снова перейти к выбору готового или пользовательского задания, что представлено на рисунке 4.3. Если пользователь выбирает свое задание для обучения, то на экране выводится инструкция по тому, как правильно вводить обучающие входные и выходные данные, что показано на рисунке 4.4. Если пользователь неправильно заполнит файл, программа выводит, что именно нужно изменить в файле, что продемонстрированно на рисунке 4.5. После чего пользователю необходимо выбрать количество эпох и скорость обучения. Спустя определенное количество эпох выводится результат тестирования, и пользователь может вводить свои задания, что показано на рисунке 4.6. Также пользователь сможет снова вернуться к выбору задания или же выйти из программы.

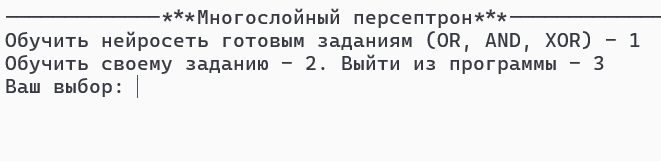


Рисунок 4.1 – Приветственная страница

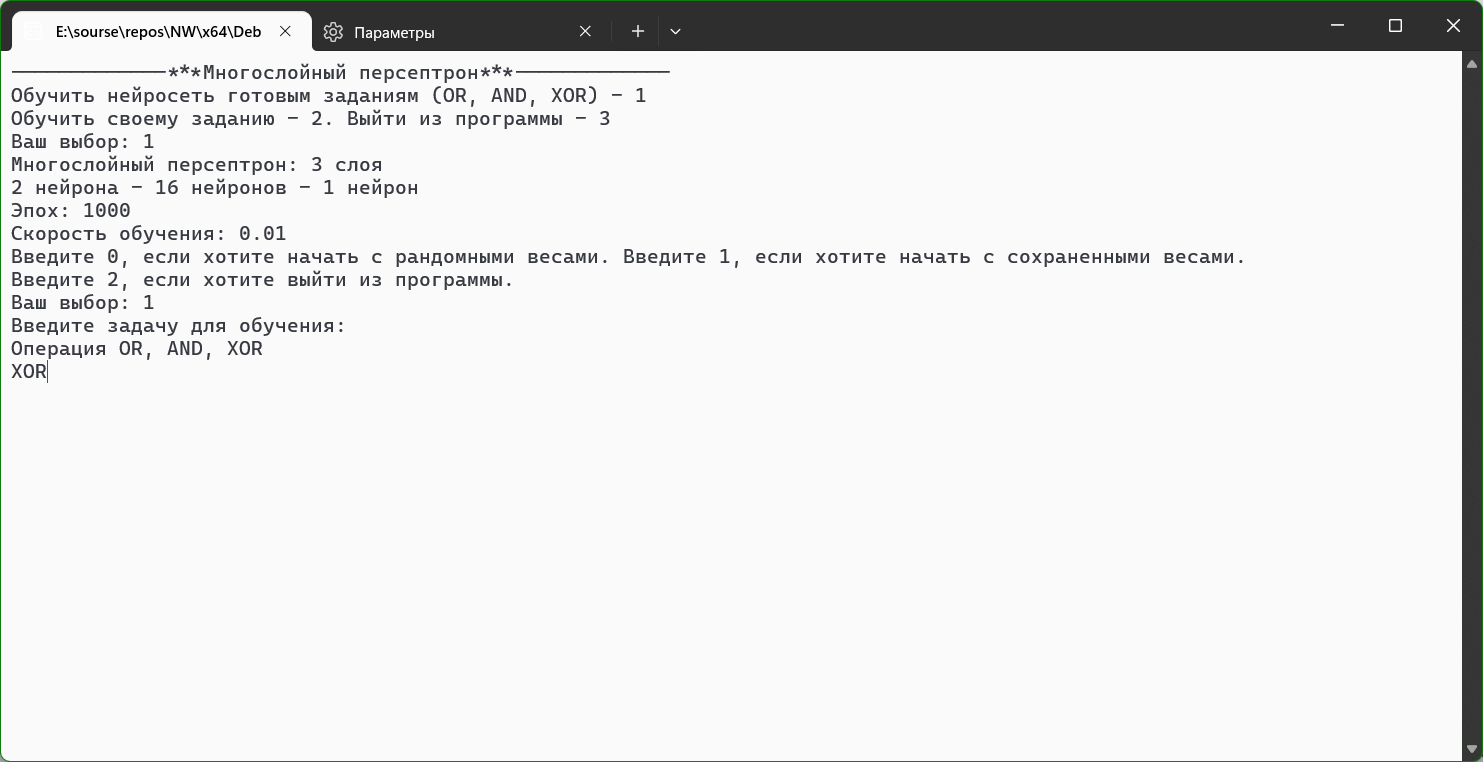


Рисунок 4.2 – Страница выбора задачи и значений весов и смещений

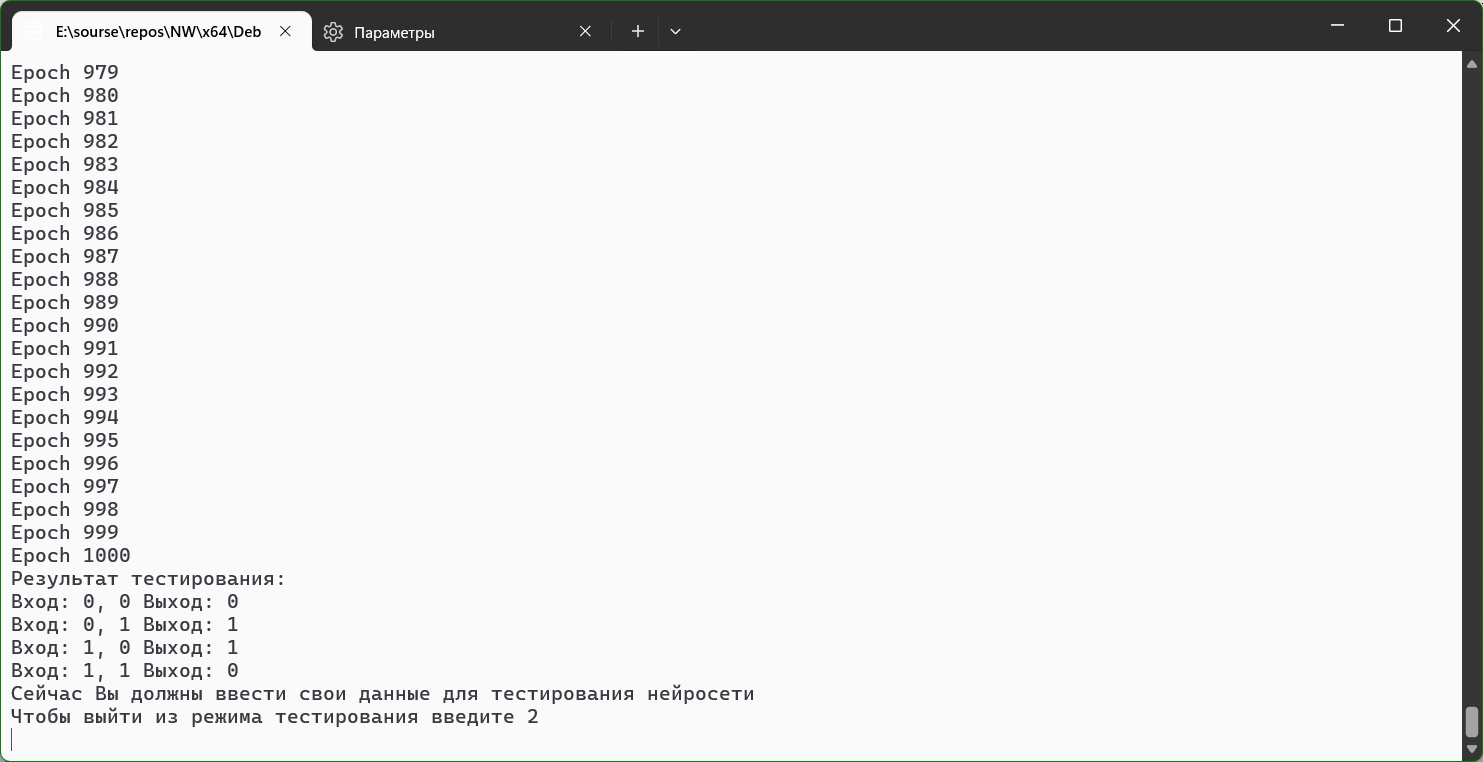


Рисунок 4.3 – Результат тестирования и ввод своих заданий

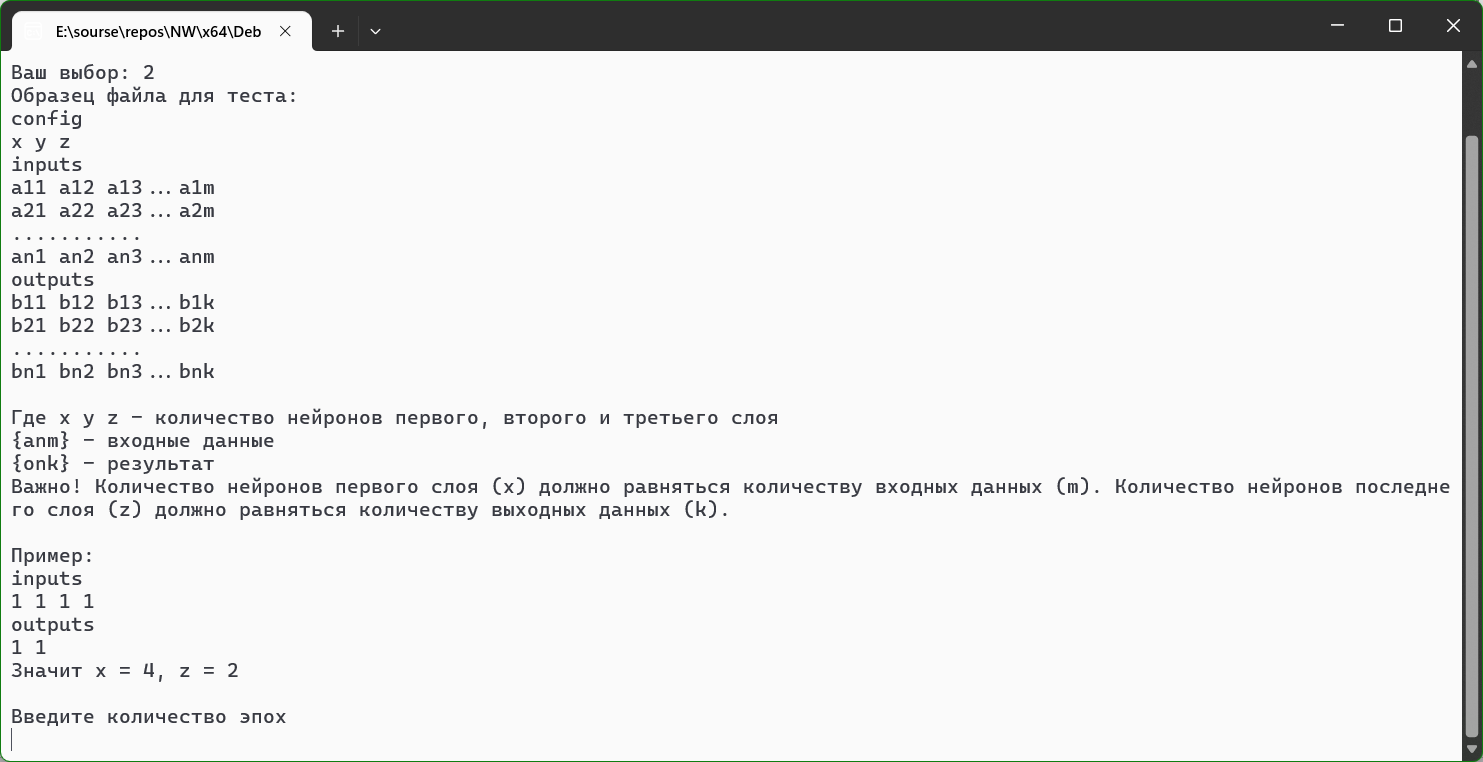


Рисунок 4.4 – Инструкция по данным обучения

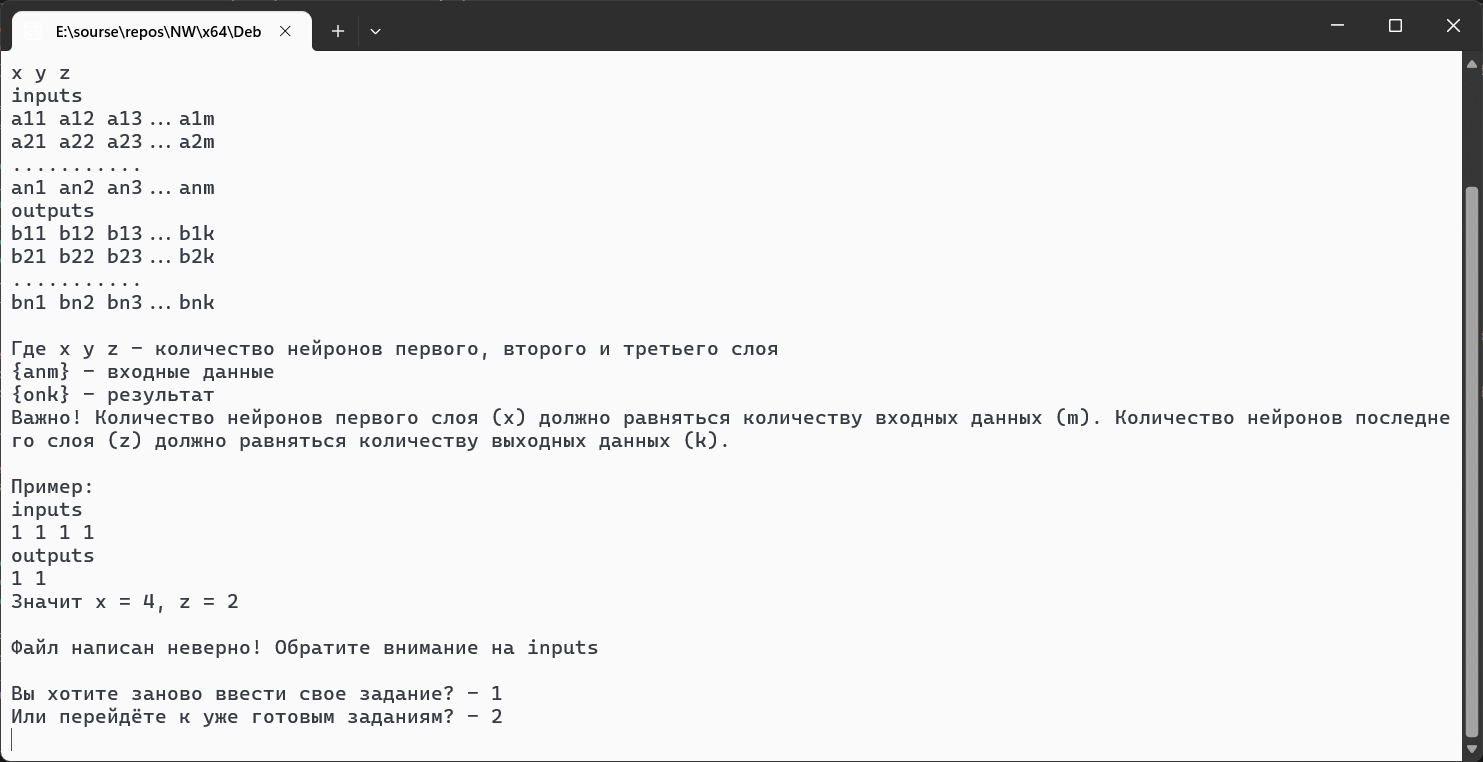


Рисунок 4.5 – Приветственная страница

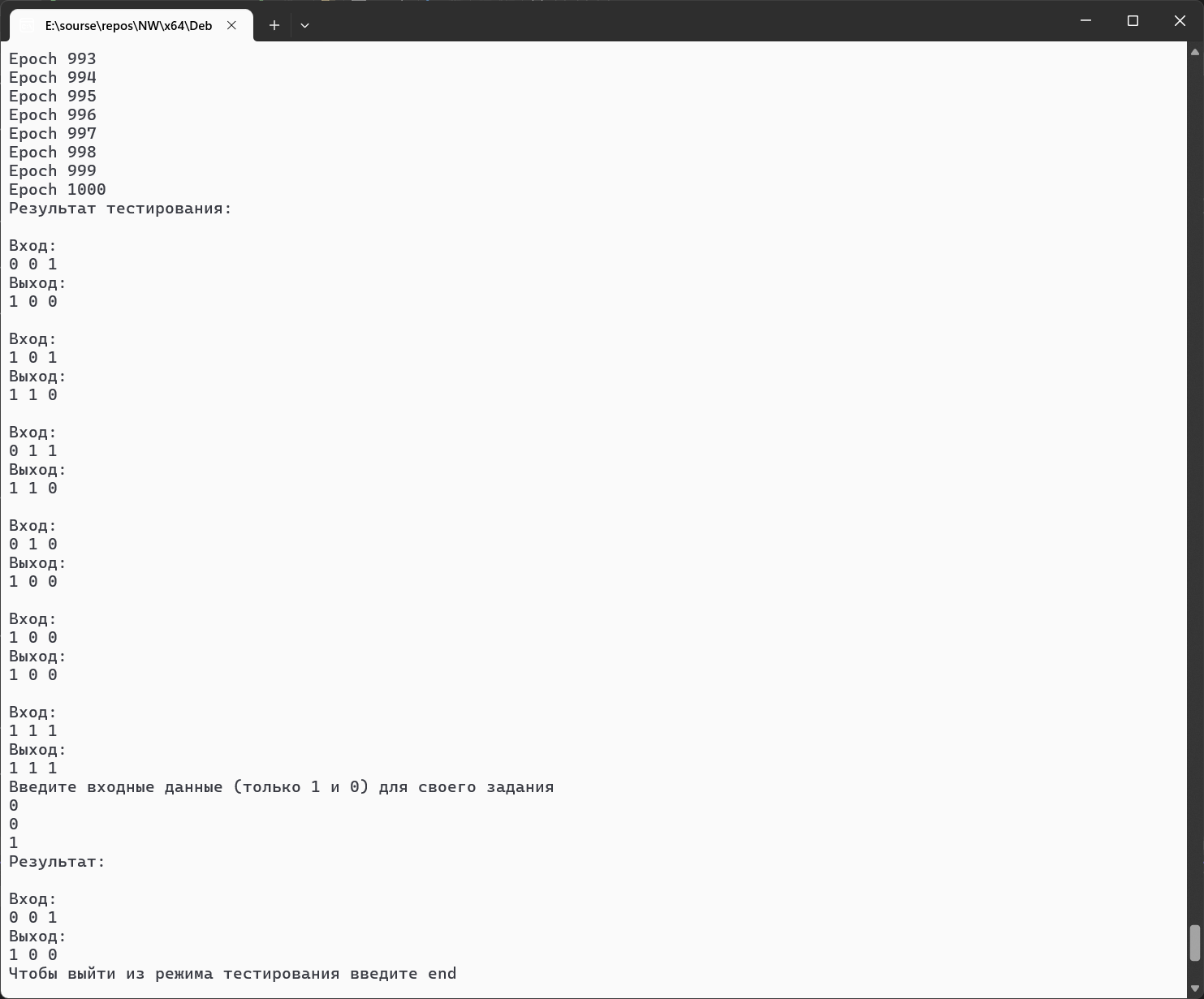


Рисунок 4.6 – Результат тестирования и ввод своих заданий

4.3 Выводы по главе

В данной главе было произведено последовательное знакомство с разработанным функционалом многослойного персептрона. Так же была проведена демонстрация работы программы, в которую вошло: приветственное окно программы, окно тестирований и ввода своих заданий, обучение, инструкция по входным и выходным данным для обучения, меню выбора заданий, меню выбора весов и смещений, проверка файла с данными для обучения.

Заключение

В ходе выполнения данной работы была разработана модель многослойного персептрона, которое позволяет пользователям быстро получать результат любого задания с помощью использования простой сетки нейронной искусственной сети, а также алгоритмов обучения. Программа отвечает всем установленным требованиям.

Данное программное средство полностью реализует функции, поставленные в пункте 1.3, а именно:

* выбор вида обучения;
* выбор задачи для обучения;
* настройка количества нейронов, эпох и скорости обучения;
* отображение результатов тестирования после обучения;
* сохранение весов и смещений;
* механизм заполнения случайными числами весов и смещений;
* выбор начальных весов и смещений;
* ввод заданий для обученной нейросети.

Получены и закреплены навыки разработки и создания программного средства, связанные с использованием языка С++.

Перечень использованных информационных ресурсов

1. Многослойный персептрон (Multilayered perceptron) [Электронный ресурс], URL: <https://wiki.loginom.ru/articles/multilayered-perceptron.html> (дата обращения 20.06.2024).
2. Нейросеть [Электронный ресурс], URL: <https://blog.skillfactory.ru/glossary/nejronnaya-set/> (дата обращения 20.06.2024).
3. Виды нейронных сетей [Электронный ресурс], URL: <https://iis.guu.ru/blog/vidy-neironnih-setey/> (дата обращения 20.06.2024).
4. Библиотека iostream [Электронный ресурс], URL: <https://ru.wikipedia.org/wiki/Iostream> (дата обращения 20.06.2024).
5. Библиотека windows.h [Электронный ресурс], URL: <https://en.wikipedia.org/wiki/Windows.h> (дата обращения 20.06.2024).
6. Библиотека random [Электронный ресурс], URL: <https://learn.microsoft.com/ru-ru/cpp/standard-library/random?view=msvc-170> (дата обращения 20.06.2024).
7. Библиотека fstream [Электронный ресурс], URL: <https://ru.wikipedia.org/wiki/Fstream> (дата обращения 20.06.2024).
8. Библиотека sstream [Электронный ресурс], URL: <https://ru.wikipedia.org/wiki/Sstream> (дата обращения 20.06.2024).
9. Нейронная сеть с прямой связью [Электронный ресурс], URL: <https://ru.wikipedia.org/wiki/Нейронная_сеть_с_прямой_связью#Однослойный_персептрон> (дата обращения 20.06.2024).
10. Многослойный перцептрон и алгоритм обратного распространения ошибки [Электронный ресурс], URL: <https://www.mql5.com/ru/articles/8908>(дата обращения 20.06.2024).
11. Visual Studio [Электронный ресурс], URL: <https://en.wikipedia.org/wiki/Visual_Studio> (дата обращения 20.06.2024).

Приложение А Листинг программы

Листинг А.1 – Заголовок network.h

#pragma once

#ifndef NETWORK\_H

#define NETWORK\_H

#include <iostream>

#include <random>

#include <windows.h>

#include <fstream>

#include <string>

#include <vector>

#include <ctime>

#include <cmath>

#include <sstream>

#include <cstdlib>

using namespace std;

double modrelu(double x);

double modrelu\_derivative(double x);

vector<double> matrix\_multiply\_sum(const vector<vector<double>>& A, const vector<double> & B, int input\_neuro, int output\_neuro, const vector<double>& bias);

class MultilayerPerceptron {

int input\_size;

int hidden\_size;

int output\_size;

vector<vector<double>> weights\_firstL;

vector<vector<double>> weights\_secondL;

vector<double> bias\_firstL;

vector<double> bias\_secondL;

public:

vector<double> hidden\_output;

MultilayerPerceptron(int input\_size, int hidden\_size, int output\_size) {

this->input\_size = input\_size; this->hidden\_size = hidden\_size; this->output\_size = output\_size;

};

void Init(int n);

vector<double> forward\_propagation(const vector<double>& input, vector<double>& hidden\_output);

void backpropagation\_updating\_weights(const vector<double>& output, const vector<double>& target, double learning\_rate, const vector<double>& input);

vector<double> matrix\_multiply(const vector<vector<double>>& A, const vector<double>& B, int input\_neuro, int output\_neuro, const vector<double>& input);

void train(const vector<vector<double>>& inputs, const vector<vector<double>>& targets, int epochs, double learning\_rate, string ans);

void file(string ans);

};

void clean();

void read\_targets(vector<vector<double>>& targets\_or, vector<vector<double>>& targets\_and, vector<vector<double>>& targets\_xor);

void tasks(string a, int l, vector<vector<double>>& inputs\_or, vector<vector<double>>& inputs\_and,

vector<vector<double>>& inputs\_xor, vector<vector<double>>& targets\_or,

vector<vector<double>>& targets\_and, vector<vector<double>>& targets\_xor, MultilayerPerceptron& mlp, string ans);

void read\_conf(vector<int>& config);

void read\_inputs(vector<vector<double>>& inputs\_or, vector<vector<double>>& inputs\_and, vector<vector<double>>& inputs\_xor);

bool read\_test(vector<int>& config, vector<vector<double>>& inputs\_other, vector<vector<double>>& targets\_other);

bool check(string s);

void task\_test(int v, double L, string ans, vector<vector<double>>& inputs\_other, vector<vector<double>>& targets\_other, MultilayerPerceptron& MLP);

#endif

Листинг А.2 – network.cpp

#include "network.h"

double modrelu(double x) {

if (x < 0) return 0;

else if (x > 1) return 1;

else return x;

}

double modrelu\_derivative(double x) {

if (x < 0 || x > 1) return 0;

else return 1;

}

vector<double> matrix\_multiply\_sum(const vector<vector<double>>& A, const vector<double>& B, int input\_neuro, int output\_neuro, const vector<double>& bias) {

vector<double> C;

for (int i = 0; i < output\_neuro; i++) {

double temp = 0;

for (int j = 0; j < input\_neuro; j++) {

temp += A[i][j] \* B[j];

}

temp += bias[i];

C.push\_back(temp);

}

return C;

}

void MultilayerPerceptron::Init(int n)

{

if (n == 0) {

random\_device rd;

mt19937 gen(rd());

uniform\_real\_distribution<> dist(0.0, 1.0);

for (int i = 0; i < hidden\_size; i++) {

vector<double> row;

for (int j = 0; j < input\_size; j++) {

row.push\_back(dist(gen));

}

weights\_firstL.push\_back(row);

}

for (int i = 0; i < output\_size; i++) {

vector<double> row;

for (int j = 0; j < hidden\_size; j++) {

row.push\_back(dist(gen));

}

weights\_secondL.push\_back(row);

}

for (int i = 0; i < hidden\_size; i++) bias\_firstL.push\_back(dist(gen));

for (int i = 0; i < output\_size; i++) bias\_secondL.push\_back(dist(gen));

}

else {

ifstream fw1("WEIGHTFIRST.txt");

ifstream fw2("WEIGHTSECOND.txt");

ifstream fb1("BFIRST.txt");

ifstream fb2("BSECOND.txt");

if (!fw1.is\_open() || !fw2.is\_open() || !fb1.is\_open() || !fb2.is\_open()) { cout << "Error file!\n"; return; }

string s;

for (int i = 0; i < hidden\_size; i++) {

vector<double> row;

for (int j = 0; j < input\_size; j++) {

getline(fw2, s);

const char\* q = s.c\_str();

row.push\_back(atof(q));

}

weights\_firstL.push\_back(row);

}

for (int i = 0; i < output\_size; i++) {

vector<double> row;

for (int j = 0; j < hidden\_size; j++) {

getline(fw1, s);

const char\* c = s.c\_str();

row.push\_back(atof(c));

}

weights\_secondL.push\_back(row);

}

for (int i = 0; i < hidden\_size; i++) { getline(fb1, s); bias\_firstL.push\_back(stod(s)); }

for (int i = 0; i < output\_size; i++) {

getline(fb2, s); bias\_secondL.push\_back(stod(s));

}

}

}

vector<double> MultilayerPerceptron::forward\_propagation(const vector<double>& input, vector<double> &hidden\_output)

{

hidden\_output = matrix\_multiply\_sum(weights\_firstL, input, input\_size, hidden\_size, bias\_firstL);

for (int i = 0; i < hidden\_output.size(); i++) hidden\_output[i] = modrelu(hidden\_output[i]);

vector<double> output = matrix\_multiply\_sum(weights\_secondL, hidden\_output, hidden\_size, output\_size, bias\_secondL);

for (int i = 0; i < output.size(); i++) output[i] = modrelu(output[i]);

return output;

}

void MultilayerPerceptron::backpropagation\_updating\_weights(const vector<double>& output, const vector<double>& target, double learning\_rate, const vector<double>& input)

{

int c = 0;

vector<double> output\_errors;

for (int i = 0; i < output\_size; i++) {

output\_errors.push\_back((target[i] - output[i]) \* modrelu\_derivative(output[i]));

}

vector<double> hidden\_errors = matrix\_multiply(weights\_secondL, output\_errors, hidden\_size, output\_size, hidden\_output);

for (int i = 0; i < output\_size; i++) {

for (int j = 0; j < hidden\_size; j++) {

weights\_secondL[i][j] += hidden\_output[j] \* output\_errors[i] \* learning\_rate;

}

bias\_secondL[i] += learning\_rate \* output\_errors[i];

}

for (int i = 0; i < hidden\_size; i++) {

for (int j = 0; j < input\_size; j++) {

weights\_firstL[i][j] += learning\_rate \* hidden\_errors[i] \* input[j];

}

bias\_firstL[i] += learning\_rate \* hidden\_errors[i];

}

}

vector<double> MultilayerPerceptron::matrix\_multiply(const vector<vector<double>>& A, const vector<double>& B, int input\_neuro, int output\_neuro, const vector<double>& input)

{

vector<double> C;

for (int i = 0; i < input\_neuro; ++i) {

double temp = 0;

for (int j = 0; j < output\_size; ++j) {

temp += A[j][i] \* B[j];

}

temp \*= modrelu\_derivative(input[i]);

C.push\_back(temp);

}

return C;

}

void MultilayerPerceptron::train(const vector<vector<double>>& inputs, const vector<vector<double>>& targets, int epochs, double learning\_rate, string ans)

{

for (int epoch = 0; epoch < epochs; epoch++) {

for (int i = 0; i < inputs.size(); i++) {

vector<double> a = forward\_propagation(inputs[i], hidden\_output);

backpropagation\_updating\_weights(a, targets[i], learning\_rate, inputs[i]);

}

cout << "Epoch " << epoch + 1 << endl;

}

file(ans);

}

void MultilayerPerceptron::file(string ans)

{

if (ans == "1") {

ofstream fw1("WEIGHTFIRST.txt");

ofstream fw2("WEIGHTSECOND.txt");

ofstream fb1("BFIRST.txt");

ofstream fb2("BSECOND.txt");

if (!fw1.is\_open() || !fw2.is\_open() || !fb1.is\_open() || !fb2.is\_open()) { cout << "Error file!\n"; return; }

for (auto& weight : weights\_firstL) {

for (int i = 0; i < weight.size(); i++) {

fw1 << weight[i]; fw1 << endl;

}

}

for (auto& weight : weights\_secondL) {

for (int i = 0; i < weight.size(); i++) {

fw2 << weight[i]; fw2 << endl;

}

}

for (int i = 0; i < bias\_firstL.size(); i++) {

fb1 << bias\_firstL[i]; fb1 << endl;

}

for (int i = 0; i < bias\_secondL.size(); i++) {

fb2 << bias\_secondL[i]; fb2 << endl;

}

}

else if (ans == "2") {

ofstream fw1("WEIGHTFIRSTTEST.txt");

ofstream fw2("WEIGHTSECONDTEST.txt");

ofstream fb1("BFIRSTTEST.txt");

ofstream fb2("BSECONDTEST.txt");

if (!fw1.is\_open() || !fw2.is\_open() || !fb1.is\_open() || !fb2.is\_open()) { cout << "Error file!\n"; return; }

for (auto& weight : weights\_firstL) {

for (int i = 0; i < weight.size(); i++) {

fw1 << weight[i]; fw1 << endl;

}

}

for (auto& weight : weights\_secondL) {

for (int i = 0; i < weight.size(); i++) {

fw2 << weight[i]; fw2 << endl;

}

}

for (int i = 0; i < bias\_firstL.size(); i++) {

fb1 << bias\_firstL[i]; fb1 << endl;

}

for (int i = 0; i < bias\_secondL.size(); i++) {

fb2 << bias\_secondL[i]; fb2 << endl;

}

}

}

void clean() {

cin.clear();

cin.ignore((numeric\_limits<streamsize>::max)(), '\n');

cout << "Error!" << endl;

}

void read\_inputs(vector<vector<double>>& inputs\_or, vector<vector<double>>& inputs\_and, vector<vector<double>>& inputs\_xor) {

ifstream inp("testinput.txt");

string s, str;

double w;

const char\* q;

int c = 0;

if (!inp.is\_open()) { cout << "Error file!\n"; return; }

while (getline(inp, s)){

if (s == "inputs\_or" || s == "inputs\_and" || s == "inputs\_xor") continue;

else {

vector<double> temp;

stringstream ss(s);

while (getline(ss, str, ' ')) {

q = str.c\_str();

w = atof(q); temp.push\_back(w);

}

c++;

if (c <= 4) inputs\_or.push\_back(temp);

else if(c>4 && c<=8) inputs\_and.push\_back(temp);

else if(c>8) inputs\_xor.push\_back(temp);

}

}

}

void read\_targets(vector<vector<double>>& targets\_or, vector<vector<double>>& targets\_and, vector<vector<double>>& targets\_xor) {

ifstream tar("target.txt");

string s;

double w;

const char\* q;

int c = 0;

if (!tar.is\_open()) {

cout << "Error file!\n";return;

}

while (getline(tar, s)) {

if (s == "targets\_or" || s == "targets\_and" || s == "targets\_xor") continue;

else {

vector<double> temp;

q = s.c\_str();

w = atof(q); temp.push\_back(w);

c++;

if (c <= 4) targets\_or.push\_back(temp);

else if (c > 4 && c <= 8) targets\_and.push\_back(temp);

else if (c > 8) targets\_xor.push\_back(temp);

}

}

}

void read\_conf(vector<int> &config) {

ifstream conf("testconfig.txt");

if (!conf.is\_open()) {

cout << "Error file!\n"; return ;

}

string s, st;

const char\* q;

int c;

vector <int> temp;

while (getline(conf, s)) {

stringstream ss(s);

while (getline(ss, st, ' ')) {

q = st.c\_str();

c = stoi(q); temp.push\_back(c);

}

}

config = temp;

}

bool read\_test(vector<int> &config, vector<vector<double>> &inputs\_other, vector<vector<double>> & targets\_other) {

cout << "Образец файла для теста:\nconfig\nx y z\ninputs\na11 a12 a13...a1m\na21 a22 a23...a2m\n...........\nan1 an2 an3...anm";

cout << "\noutputs\nb11 b12 b13...b1k\nb21 b22 b23...b2k\n...........\nbn1 bn2 bn3...bnk\n";

cout << "\nГде x y z - количество нейронов первого, второго и третьего слоя\n{anm} - входные данные\n{onk} - результат\nВажно! Количество нейронов первого слоя (x) должно равняться количеству ";

cout << "входных данных (m). Количество нейронов последнего слоя (z) должно равняться количеству выходных данных (k).\n\nПример:\ninputs\n1 1 1 1\noutputs\n1 1\nЗначит x = 4, z = 2\n\n";

ifstream conf("test.txt");

if (!conf.is\_open()) {

cout << "Error file!\n"; return false;

}

if (conf.peek() == EOF) {

cout << "Файл пустой!\n"; return false;

}

vector<vector<double>> inputs1\_other;

vector<vector<double>> outputs1\_other;

string s, st;

vector<int> temp;

int c = 0, k = 0;

const char\* q;

int w;

bool f = false; int u = 0;

bool fi = false;

while (getline(conf, s)) {

if (c == 0 && s != "config") { cout << "Файл написан неверно! Обратите внимание на config\n"; return false; }

else if (c == 0 && s == "config") {

c++; continue;

}

//проверка количества нейронов

if (c == 1) {

if (k == 3) {

cout << "Слоёв всего три!\n"; return false;

}

if(s == "") {

cout << "Что-то не так с количеством нейронов!\n"; return false;

}

stringstream ss(s);

while (getline(ss, st, ' ')) {

if (!(check(st))) { cout << "Количество нейронов должно быть числом (>0)\n"; return false; }

q = st.c\_str();

w = stoi(q);

if (w == 0) {

cout << "Количество нейронов должно быть числом (>0)\n"; return false;

}

temp.push\_back(w);

k++;

}

}

if(k < 3) {

cout << "Слоёв меньше трёх!\n"; return false;

}

if (c == 2 && s != "inputs") { cout << "Файл написан неверно! Обратите внимание на inputs\n"; return false; }

else if (c == 2 && s == "inputs") {

c++; fi = true; continue;

}

if ((c == 3 || c == 4 || c == 5) && s == "outputs") { cout << "Должно быть более двух примеров\n"; return false; }

double w1;

vector<double> tmp;

stringstream sss(s);

if (s != "outputs" && f == false && fi == true) {

while (getline(sss, st, ' ')) {

if (!(check(st))) { cout << "Входные данные - числа (только 1 и 0)\n"; return false; }

q = st.c\_str();

w1 = atof(q);

if (!(w1 == 0 || w1 == 1)) { cout << "Входные данные - числа (только 1 и 0)\n"; return false; }

tmp.push\_back(w1);

}

inputs1\_other.push\_back(tmp);

}

else if (s == "outputs") {

c++; u++; f = true;

if (u > 1) { cout << "Несколько раз outputs\n"; return false; }continue;

}

if (f == true && fi == true) {

vector<double> tm;

stringstream ssss(s);

while (getline(ssss, st, ' ')) {

if (!(check(st))) { cout << "Выходные данные - числа (только 1 и 0)\n"; return false; }

q = st.c\_str();

w1 = atof(q);

if (!(w1 == 0 || w1 == 1)) { cout << "Выходные данные - числа (только 1 и 0)\n"; return false; }

tm.push\_back(w1);

}

outputs1\_other.push\_back(tm);

}

else if(f == true && fi == false){

cout << "В файле ошибка!\n"; return false;

}

c++;

}

if (f == false) {

cout << "Нет outputs\n"; return false;

}

if (outputs1\_other.size() != inputs1\_other.size()) { cout << "Количество тестов входных и выходных данных должно быть одинаковым\n"; return false; }

vector<double> temps;

for (auto& input : inputs1\_other) {

int h = 0;

for (int i = 0; i < input.size(); i++) {

h++;

}

temps.push\_back(h);

}

float sum = 0;

for (int i = 0; i < temps.size(); i++) {

if (temps[i] != temp[0]) { cout << " Количество нейронов первого слоя (x) должно равняться количеству входных данных (m).\n"; return false; }

sum += temps[i];

}

if(sum/ temps.size() != temps[0]) { cout << " Количество чисел в inputs на каждой строке отличается.\n"; return false; }

vector<double> te;

for (auto& output : outputs1\_other) {

int h1 = 0;

for (int i = 0; i < output.size(); i++) {

h1++;

}

te.push\_back(h1);

}

float sum1 = 0;

for (int i = 0; i < te.size(); i++) {

if (te[i] != temp[2]) { cout << " Количество нейронов последнего слоя (z) должно равняться количеству выходных данных (k).\n"; return false; }

sum1 += te[i];

}

if (sum1 / te.size() != te[0]) { cout << " Количество чисел в outputs на каждой строке отличается.\n"; return false; }

if(f==false) { cout << "Нет outputs\n"; return false; }

config = temp; inputs\_other = inputs1\_other; targets\_other = outputs1\_other;

return true;

}

void task\_test(int v, double L, string ans, vector<vector<double>>& inputs\_other, vector<vector<double>>& targets\_other, MultilayerPerceptron& MLP) {

MLP.Init(0);

MLP.train(inputs\_other, targets\_other, v, L, ans);

cout << "Результат тестирования:" << endl;

int c = 0;

double q;

for (auto& input : inputs\_other) {

vector<double> output = MLP.forward\_propagation(input, MLP.hidden\_output);

c = input.size();

cout << "\nВход:\n";

for (int i = 0; i < input.size(); i++) cout << input[i] << " ";

cout << "\nВыход:\n";

for (int i = 0; i < output.size(); i++) cout << round(output[i]) << " ";

cout << endl;

}

string z = "";

while (z != "end") {

vector <double>input\_test;

cout << "Введите входные данные (только 1 и 0) для своего задания\n";

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

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1)) {

clean();

cout << "только 1 и 0\n"; cin >> q;

}

input\_test.push\_back(q);

}

cout << "Результат:" << endl;

vector<double> output = MLP.forward\_propagation(input\_test, MLP.hidden\_output);

cout << "\nВход:\n";

for (int i = 0; i < input\_test.size(); i++) cout << input\_test[i] << " ";

cout << "\nВыход:\n";

for (int i = 0; i < output.size(); i++) cout << round(output[i]) << " ";

cout << endl;

cout << "Чтобы выйти из режима тестирования введите end\n"; cin >> z;

}

}

void tasks(string a, int l, vector<vector<double>>& inputs\_or, vector<vector<double>>& inputs\_and,

vector<vector<double>>& inputs\_xor, vector<vector<double>>& targets\_or,

vector<vector<double>>& targets\_and, vector<vector<double>>& targets\_xor, MultilayerPerceptron& mlp, string ans)

{

double q, w;

if (a == "OR") {

mlp.Init(l);

mlp.train(inputs\_or, targets\_or, 1000, 0.01, ans);

cout << "Результат тестирования:" << endl;

for (auto& input : inputs\_or) {

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

}

cout << "Сейчас Вы должны ввести свои данные для тестирования нейросети\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор OR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор OR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

while (!(q == 2 || w == 2)) {

cout << "Результат:" << endl;

vector<double> input; input.push\_back(q); input.push\_back(w);

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

cout << "Введите два числа (только 1 и 0), к которым примените оператор OR\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор OR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор OR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

}

}

else if (a == "AND") {

mlp.Init(l);

mlp.train(inputs\_and, targets\_and, 1000, 0.01, ans);

cout << "Результат тестирования:" << endl;

for (auto& input : inputs\_and) {

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

}

cout << "Сейчас Вы должны ввести свои данные для тестирования нейросети\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор AND\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор AND\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

while (!(q == 2 || w == 2)) {

cout << "Результат:" << endl;

vector<double> input; input.push\_back(q); input.push\_back(w);

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

cout << "Введите два числа (только 1 и 0), к которым примените оператор AND\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор AND\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор AND\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

}

}

else if (a == "XOR") {

mlp.Init(l);

int k = 1000;

if (l == 0) k = 7000;

mlp.train(inputs\_xor, targets\_xor, k, 0.01, ans);

cout << "Результат тестирования:" << endl;

for (auto& input : inputs\_xor) {

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

}

cout << "Сейчас Вы должны ввести свои данные для тестирования нейросети\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор XOR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор XOR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

while (!(q == 2 || w == 2)) {

cout << "Результат:" << endl;

vector<double> input; input.push\_back(q); input.push\_back(w);

vector<double> output = mlp.forward\_propagation(input, mlp.hidden\_output);

cout << "Вход: " << input[0] << ", " << input[1] << " Выход: " << round(output[0]) << endl;

cout << "Введите два числа (только 1 и 0), к которым примените оператор XOR\nЧтобы выйти из режима тестирования введите 2\n";

cin >> q;

while (!cin || cin.peek() != '\n' || !(q == 0 || q == 1 || q == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор XOR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> q;

}

if (q == 2) return;

cin >> w;

while (!cin || cin.peek() != '\n' || !(w == 0 || w == 1 || w == 2)) {

clean();

cout << "Введите числа (только 1 и 0), к которым примените оператор XOR\nЧтобы выйти из режима тестирования введите 2\n"; cin >> w;

}

}

}

}

bool check(string s) {

return !s.empty() && std::find\_if(s.begin(),

s.end(), [](unsigned char c) { return !std::isdigit(c); }) == s.end();

}

Листинг А.2 –main.cpp

#include "network.h"

#include <iostream>

using namespace std;

int main()

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

string ans;

cout << "-------------\*\*\*Многослойный персептрон\*\*\*-------------\n";

while (true) {

cout << "Обучить нейросеть готовым заданиям (OR, AND, XOR) - 1\nОбучить своему заданию - 2. Выйти из программы - 3\n";

cout << "Ваш выбор: "; cin >> ans;

while (!(ans == "1" || ans == "2" || ans == "3")) { cout << "Обучить нейросеть готовыми заданиями (OR, AND, XOR) - 1\n Обучить своему заданию - 2. Выйти из программы - 3\n"; cout << "Ваш выбор: "; cin >> ans;

}

if (ans == "3") return 0;

if (ans == "1") {

cout << "Многослойный персептрон: 3 слоя\n2 нейрона - 16 нейронов - 1 нейрон\nЭпох: 1000\nСкорость обучения: 0.01\n";

vector<int> config;

read\_conf(config);

string a; int l;

MultilayerPerceptron mlp(config[0], config[1], config[2]);

vector<vector<double>> inputs\_or, inputs\_and, inputs\_xor;

vector<vector<double>> targets\_or, targets\_and, targets\_xor;

read\_inputs(inputs\_or, inputs\_and, inputs\_xor);

read\_targets(targets\_or, targets\_and, targets\_xor);

cout << "Введите 0, если хотите начать с рандомными весами. Введите 1, если хотите начать c сохраненными весами.\nВведите 2, если хотите выйти из программы.\n"; cout << "Ваш выбор: "; cin >> l;

while (!cin || cin.peek() != '\n' || !(l == 0 || l == 1 || l == 2)) {

clean();

cout << "Введите 0, если хотите начать с рандомных весов. Введите 1, если хотите начать c сохраненными весами.\nВведите 2, если хотите выйти из программы.\n"; cout << "Ваш выбор: "; cin >> l;

}

if (l == 2) return 0;

cout << "Введите задачу для обучения:\nОперация OR, AND, XOR\n"; cin >> a;

while (!(a == "OR" || a == "AND" || a == "XOR")) { cout << "Введите выбранную задачу (OR, AND, XOR)\n"; cin >> a; }

tasks(a, l, inputs\_or, inputs\_and, inputs\_xor, targets\_or, targets\_and, targets\_xor, mlp, ans);

}

else {

bool g = true;

vector<vector<double>> inputs\_other;

vector<vector<double>> targets\_other;

vector<int> config1;

bool fla = read\_test(config1, inputs\_other, targets\_other);

if (fla == false) {

cout << "\nВы хотите заново ввести свое задание? - 1\nИли перейдёте к уже готовым заданиям? - 2\n";

int k; cin >> k;

while (!cin || cin.peek() != '\n' || !(k == 1 || k == 2)) {

clean();

cout << "Вы хотите заново ввести свое задание? - 1\nИли перейдёте к уже готовым заданиям? - 2\n"; cin >> k;

}

if (k == 2) continue;

else {

cout << "Измените файл\n";

Sleep(3000);

fla = read\_test(config1, inputs\_other, targets\_other);

while (fla == false) {

cout << "\nВы хотите заново ввести свое задание? - 1\nИли перейдёте к уже готовым заданиям? - 2\n";

int k; cin >> k;

while (!cin || cin.peek() != '\n' || !(k == 1 || k == 2)) {

clean();

cout << "Вы хотите заново ввести свое задание? - 1\nИли перейдёте к уже готовым заданиям? - 2\n"; cin >> k;

}

if (k == 2) {

fla = true; g = false;

}

else {

cout << "Измените файл\n";

Sleep(3000);

fla = read\_test(config1, inputs\_other, targets\_other);

}

}

}

}

if (g) {

MultilayerPerceptron MLP(config1[0], config1[1], config1[2]);

int v; double L;

cout << "Введите количество эпох\n"; cin >> v;

while (!cin || cin.peek() != '\n' || v <= 0) {

clean();

cout << "Введите количество эпох(>0)\n"; cin >> v;

}

cout << "Введите скорость обучения\n"; cin >> L;

while (!cin || cin.peek() != '\n' || L <= 0) {

clean();

cout << "Введите скорость обучения(>0)\n"; cin >> L;

}

task\_test(v, L, ans, inputs\_other, targets\_other, MLP);

}

}

}}