**Программная реализация конструктора нейронных сетей**

Ганцева Е.А., Барабанов В.Ф., Гребенникова Н.И., Болдырев Д.

**Введение**

В результате бурного развития технологий нейронные сети достаточно быстро проникли практически во все сферы жизнедеятельности человека. Работа с нейронными сетями требует большое количество вычислительных ресурсов и по мере увеличения размерности сети, растут и требования к системе. Огромное количество корпораций работают над совершенствованием алгоритмов и созданием принципиально новых аппаратных комплексов. Так, например, для повышения вычислительных мощностей компьютеров часть нагрузки с центрального процессора переносится на графический ускоритель. Кроме того, современные графические адаптеры позволяют производить вычисления в параллельном режиме, что значительно увеличивает их эффективность.

В рамках данной работы осуществлена разработка программного обеспечения, которое позволит создавать нейронные сети различных типов и размеров, производить их обучение, тестирование, получать необходимые результаты. Это позволит решать большое количество практических задач, таких как задачи регрессии, классификации, распознавания образов и др.

**Обзор современных программных средств для создания и моделирования искусственных нейронных сетей**

Программные продукты для создания, обучения и тестирования нейронных сетей условно их можно разделить на несколько типов:

1) надстройки для пакетов прикладных вычислений;

2) специализированные нейросетевые пакеты;

3) универсальные нейросетевые пакеты.

Примерами являются [5]:

1)StatisticaNeuralNetworks - набор расширений,входящих в состав пакета прикладных вычисленийStatistica;

2)MatlabNeuralNetwork – набор расширений, входящих в состав пакета прикладных вычислений Matlab;

3)ExcelNeuralPackage – наборскриптов и библиотек для электронных таблиц Excel, реализующих некоторые возможности по реализации нейронных сетей.

Ко второму классу относятся программные продукты, решающие определенный, узкоспециализированный набор задач. К таким продуктам можно отнести:

1) Глаз – программа, предназначенная для выполнения обработки аэрокосмической информации;

2)NeuroshellTrader – программа, предназначенная для создания нейронных сетей и анализа рынков.

Универсальные программные пакеты имеют более специфическую направленность и функциональность, которые предназначены для работы с нейронными сетями [2].

Наиболее популярными программными продуктами являются: Neural10, NeuroSolutions, QwikNet32, MemBrain, тNeuroPro, Deductor 5.2 (Таблица 1) и др.

Таблица 1.1 – Сравнение программных продуктов

|  |  |  |
| --- | --- | --- |
| № п/п | Название программного продукта | Достоинства и недостатки |
| 1 | НейропакетNeural10 | Достоинство: является бесплатным.  Недостаток и далее: позволяется создавать только один тип сети. |
| 2 | НейропакетNeuroSolutions | Достоинства: поддерживает большое количество вариантов ввода и обработки информации.  Недостатки: высокая стоимость, отсутствие русскоязычного интерфейса. |
| 3 | Нейропакет QwikNet32 | Недостатки: реализует лишь один тип сети. |
| 4 | НейропакетMemBrain | Достоинства: бесплатен для образовательных целей, наличие мощного инструментария.  Недостатки: отсутствие русскоязычного интерфейса и справки. |
| 5 | НейропакетNeuroPro | Достоинства: имеет русскоязычный интерфейс и справку.  Недостатки: малоудобный интерфейс. |
| 6 | НейропакетDeductor 5.2 | Достоинства: бесплатная версия, русскоязычный интерфейс.  Недостатки: перегруженная функциональность. |

Проанализировав функциональные возможность различных программных продуктов, можно прийти к выводу, что все они в разной степени обладают схожими возможностями. Однако некоторые из них имеют огромное количество инструментов, которые мало подходят для знакомства с нейронными сетями, а также обладают высокой стоимостью. Другие же программные продукты слишком ограничены в наборе функций. Так, например, позволяют реализовать сети лишь определенной структуры.

Таким образом, разработка программного продукта для конструирования нейронных сетей, который прост в освоении, имеет наглядный и понятный интерфейс, является свободно распространяемой и с открытым исходным кодом, является необходимой [6].

**Функциональные требования к разрабатываемому конструктору**

**нейронных сетей**

Программный продукт должен обладать следующими функциональными возможностями:

1) создавать различные типы нейронных сетей (линейный нейрон; персептрон; многослойные сети прямого распространения);

2) настройка параметров каждого слоя, для многослойных сетей;

3) настройка параметров обучения и поддержка классических нейросетевых моделей и алгоритмов обучения;

4) сохранение параметров программы в файле настроек; загрузка параметров из файла;

5) отображение результатов работы нейронной сети в графическом виде;

6) удобная работа с входными данными (пользователь должен иметь возможность загружать готовые входные данные из файла, либо записывать эти данные вручную);

7) наличие возможности масштабирования данных (освобождает пользователя от дополнительной работы по подготовке входных или выходных данных)

8) русскоязычный интерфейс пользователя должен отвечать принципу «usability», быть наглядным, а также интуитивно понятным, что позволит пользователю сосредоточить внимание на разработке и изучении нейронных сетей;

9)наличие возможности тестировать нейронные сети;

10)доступность (программное обеспечение должно быть в свободном доступе с открытым исходным кодом, что позволит привлечь большее количество пользователей и разработчиков).

Для написания программного продукта конструирования нейронных сетей использована интегрированная среда разработки MicrosoftVisualStudio 2017.

В качестве языка программирования использован объектно-ориентированный язык C#, предназначенный для разработки приложений для платформы Microsoft. NETFramework [9].

**Программная реализация конструктора нейронных сетей**

Состав, структура и функции программного средства

Созданная программа NeuralNetConstructor предназначена для построения нейронных сетей и предоставляет возможность решать задачи регрессии, классификации и распознавании образов.

На рисунке 3.1 можно увидеть UseCaseдиаграмму конструктора нейронных сетей, на которой представлены основные функции программного обеспечения.

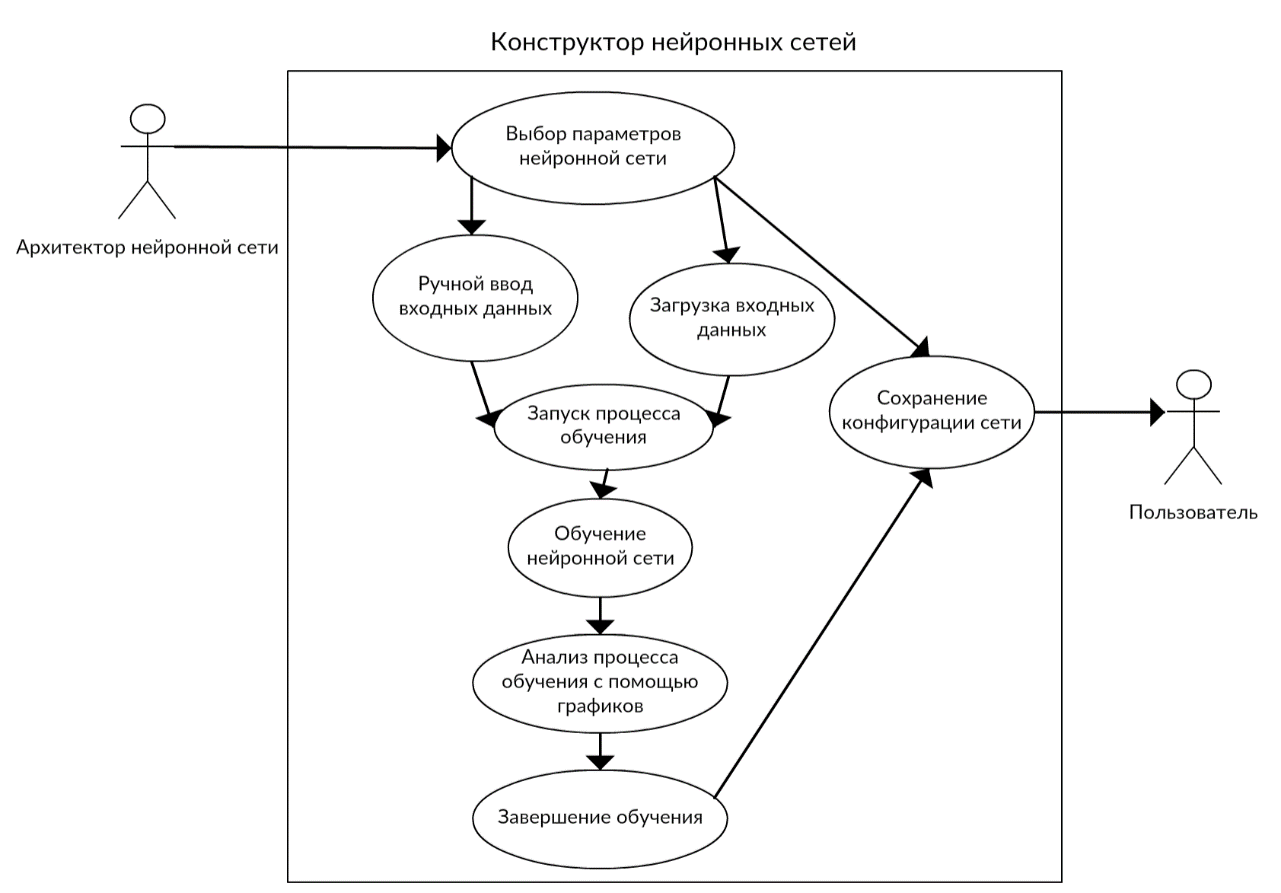


Рисунок 3.1 –UseCase диаграмма «Конструктора нейронных сетей»

На рисунке видно, что для функционирования программы со стороны пользователя необходимо передать ей настройки, к которым относится тип сети и ее конкретные параметры, а также подготовить входные данные. Входные данные могут быть загружены из файла или заданы пользователем вручную на этапе настройки параметров. В дальнейшем процесс обучения полностью скрыт от пользователя. Выходными данными (результатом обучения) являются значения весовых коэффициентов. Эти данные можно сохранить в файл, для дальнейшего использования в других проектах или для тестирования.

На рисунке 3.2 приведена архитектура программы «Конструктор нейросетей» на уровне модулей.



Рисунок 3.2 – Структура приложения на уровне модулей

Порядок работы пользователя в приложении представлен на рисунке 3.3. После ввода всех необходимых параметров происходит автоматическое создание нейронной сети. Скрывая от пользователя процесс создания сети, наличие удобного пользовательского интерфейса позволяет быстрее приступить к работе, а не тратить время на изучение документации и встроенного языка программирования. Кроме того, такой подход позволяет избежать ошибок, что значительно экономит время проектирования нейронных сетей.



Рисунок 3.3 - Порядок работы пользователя в приложении

В программе имеется возможность задания параметров для каждого слоя многослойных сетей: число нейронов, тип активационной функции, диапазон изменения весовых коэффициентов, а также коэффициентов целевой функции.

Кроме того, имеется возможность масштабирования входных и выходных данных, что позволяет пользователю создавать более гибкие модели сетей.

Для нелинейного нейрона (персептрона) список параметров схож с многослойными сетями.

Для решения задач регрессии с помощью моделей многослойных сетей и персептрона, предполагается, что имеется много входов и только один выход. Для задач классификации и распознавания образов имеется много входов и много выходов.

Для линейного нейрона список параметров значительно уже, однако не стоит считать, что линейный нейрон имеет малые вычислительные способности.

Линейный нейрон позволяет решать задачи регрессии, однако в большинстве случаев он предназначен для фильтрации сигналов путем корректировки весовых коэффициентов по методу минимизации среднеквадратической ошибки.

Для линейного нейрона можно задавать размер входа, скорость обучения и количество обучающих примеров. Так как на выходе отсутствует нелинейная активационная функция, следовательно, необязательно нормировать входные и выходные данные.

Помимо специализированных особенностей программы, которые влияют на качество работы алгоритмов, было реализовано множество функций, которые предназначены для комфортной работы пользователя.

В данном случае имеется возможность загрузки и ручного ввода данных. А также графическое отображение результатов работы. Пользователь может в режиме реального времени, на графиках, наблюдать результат работы программы, что позволяет быстро принимать решение о работоспособности созданной модели.

Разработана библиотека «Neuron.dll»,которая состоит из трех классов: Layer, Perceptron, LinearNeuron.

Процесс создания линейного нейрона происходит в классе LinearNeuronForm. При этом, первым делом, при загрузке формы создается объект класса LinearNeuron из библиотеки «Neuron.dll». В качестве параметра конструктору передается число входов нейрона. Затем вычисляется ошибка. Ошибка представляет собой разность между эталонным и текущим выходом сети. Далее ошибка передаются в качестве параметра в функцию, в которой происходит корректировка весовых коэффициентов. Обучение является итеративным процессом, поэтому данная операция будет выполняться до тех пор, пока выход сети не будет удовлетворять заданной точности. Кроме выхода сети на каждой итерации обучения вычисляется среднеквадратическая ошибка и текущая ошибка. Эти данные необходимы для построения графиков. Построение точек графиков происходит на каждой итерации обучения. После завершения обучения сформированный массив весовых коэффициентов сохраняется в классе AppSetting.

Для сохранения настроек конфигурации нейронной сети необходимо провести сериализацию класса AppSettings. Результатом данного процесса является xml файл надиске. Логично предположить, что для загрузки настроек необходимо произвести обратную операцию под названием десериализация. В общем смысле процесс проектирования программы для линейного и нелинейного нейрона является одинаковым. На рисунке 3.7 можно наблюдать порядок действий при создании многослойных сетей.



Рисунок 3.7 – Порядок работы при создании многослойной сети

Для тестирования уже обученной сети, необходимо загрузить тестовые наборы данных, создать объект нейрона или многослойной сети и подать данные на вход. Уже настроенные весовые коэффициенты находятся в классе AppSettings.

На рисунке 3.8 приведен порядок действий при тестировании обученной нейронной сети.



Рисунок 3.8 – Порядок работы при тестировании обученной сети

**Реализация интерфейсных возможностей программного средства**

При запуске программы на экране пользователя появляется главная форма - «Конструктор нейронных сетей». На рисунке 3.9 представлен внешний вид этой формы.

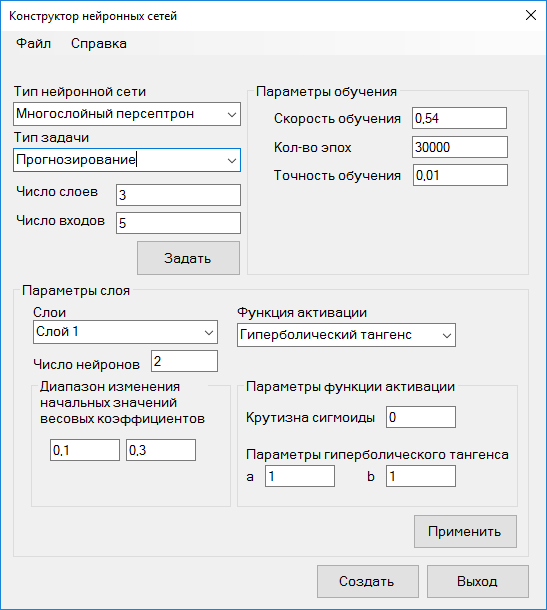


Рисунок 3.9 – Главная форма «Конструктор нейронных сетей»

На данной форме можно задать тип сети, установить параметры слоя для многослойных сетей, установить скорость обучения, число входных наборов и точность обучения.

После задания всех параметров, по нажатию на кнопке «Создать» открывается новая форма. В зависимости от параметров, которые были указаны в главной форме, создаются различные по функциональности формы.

Для управления обучением линейного нейрона предназначена форма «Линейный нейрон» показанная на рисунке 3.10.

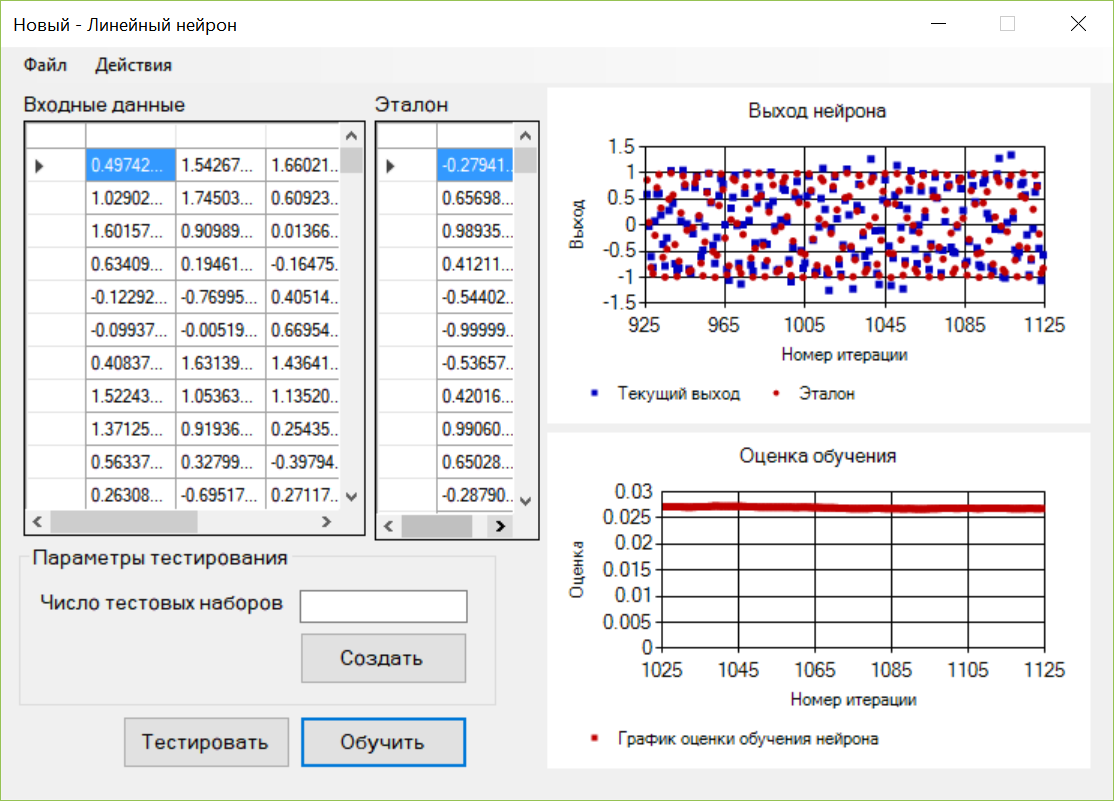


Рисунок 3.10 – Внешний вид формы «Линейный нейрон»

В рабочей области формы «Линейный нейрон» расположены две таблицы, в которые загружаются данные для обучений и тестирования. Наличие возможности визуального доступа к данным позволяет изменять их в динамическом режиме. Также на форме расположены графики для визуального отображения результатов работы процессов обучения и тестирования.

Наиболее ценной возможностью программы является возможность обучения и тестирования многослойных сетей. Для этого служит форма, показанная на рисунке 3.11.

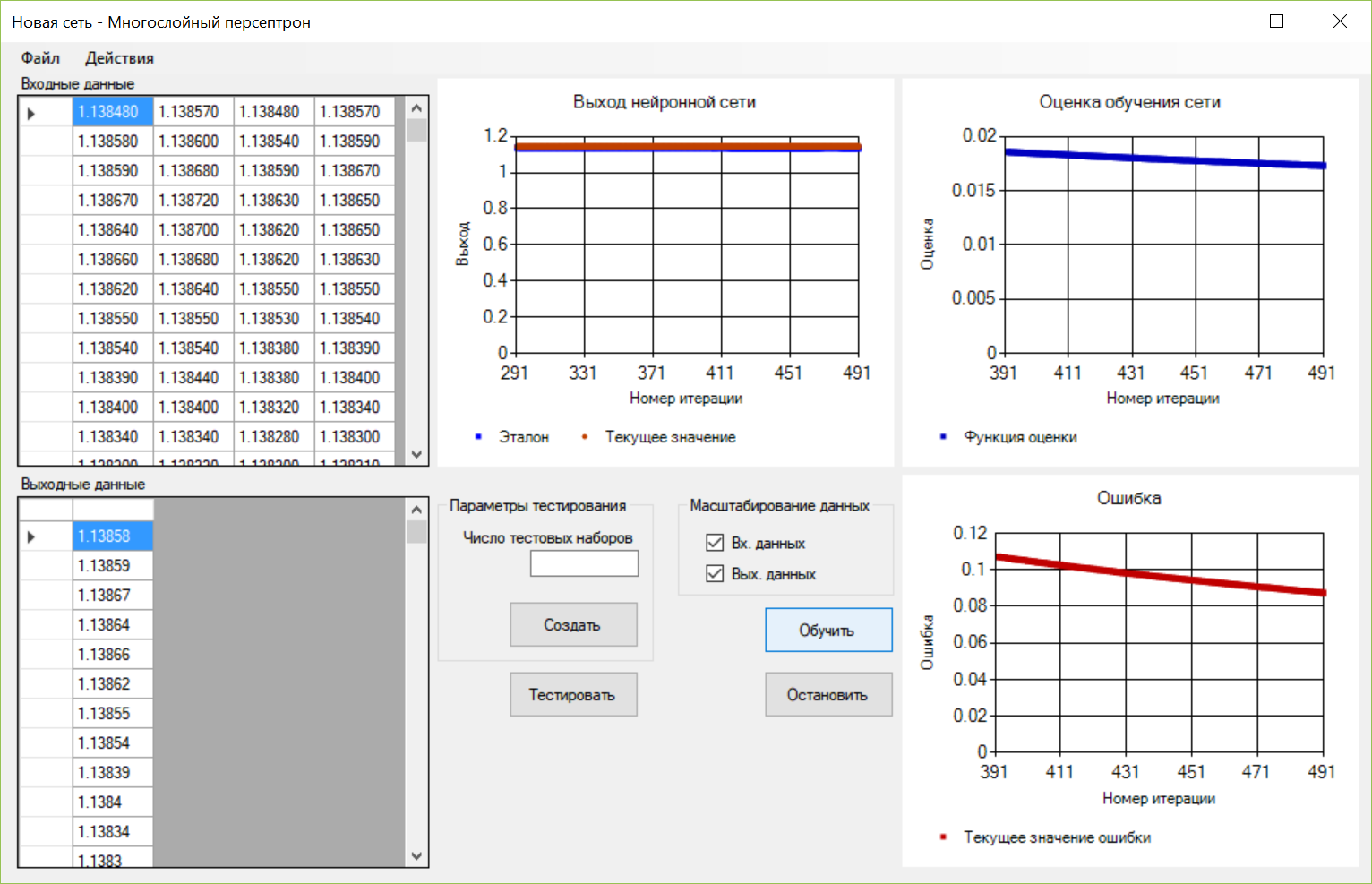


Рисунок 3.11 – Внешний вид формы «Многослойный персептрон»

На данной форме расположены идентичные элементы управления, за исключением, элементов, которые позволяют установкой флажка сообщить программе о том, что необходимо произвести модификацию входных (выходных) данных – их масштабирование.

На рисунке 3.12 показан внешний вид формы «Нелинейный нейрон».

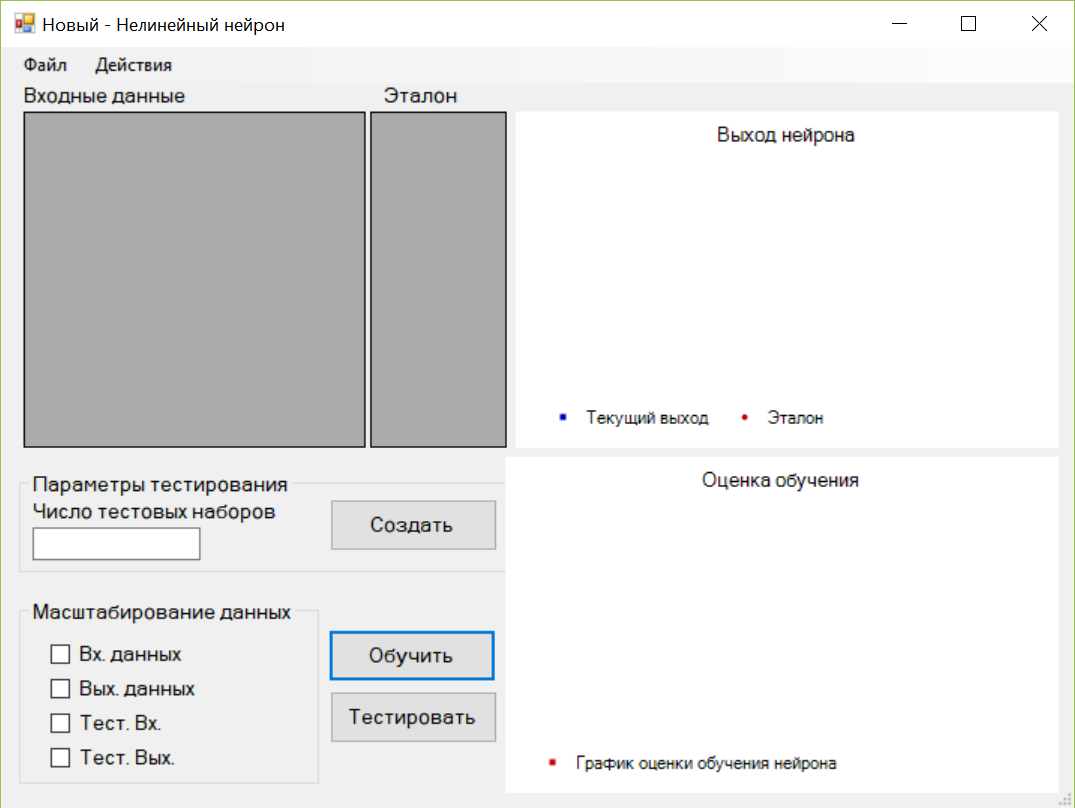


Рисунок 3.12 – Внешний вид формы «Нелинейный нейрон»

Данная форма мало чем отличается от формы «Линейный нейрон» как по внешнему виду, так и по функциональности, за исключением наличия возможности масштабирования данных.

Следующая форма предназначена для работы с многослойными сетями, решающих задачи классификации. Внешний вид показан на рисунке 3.13.

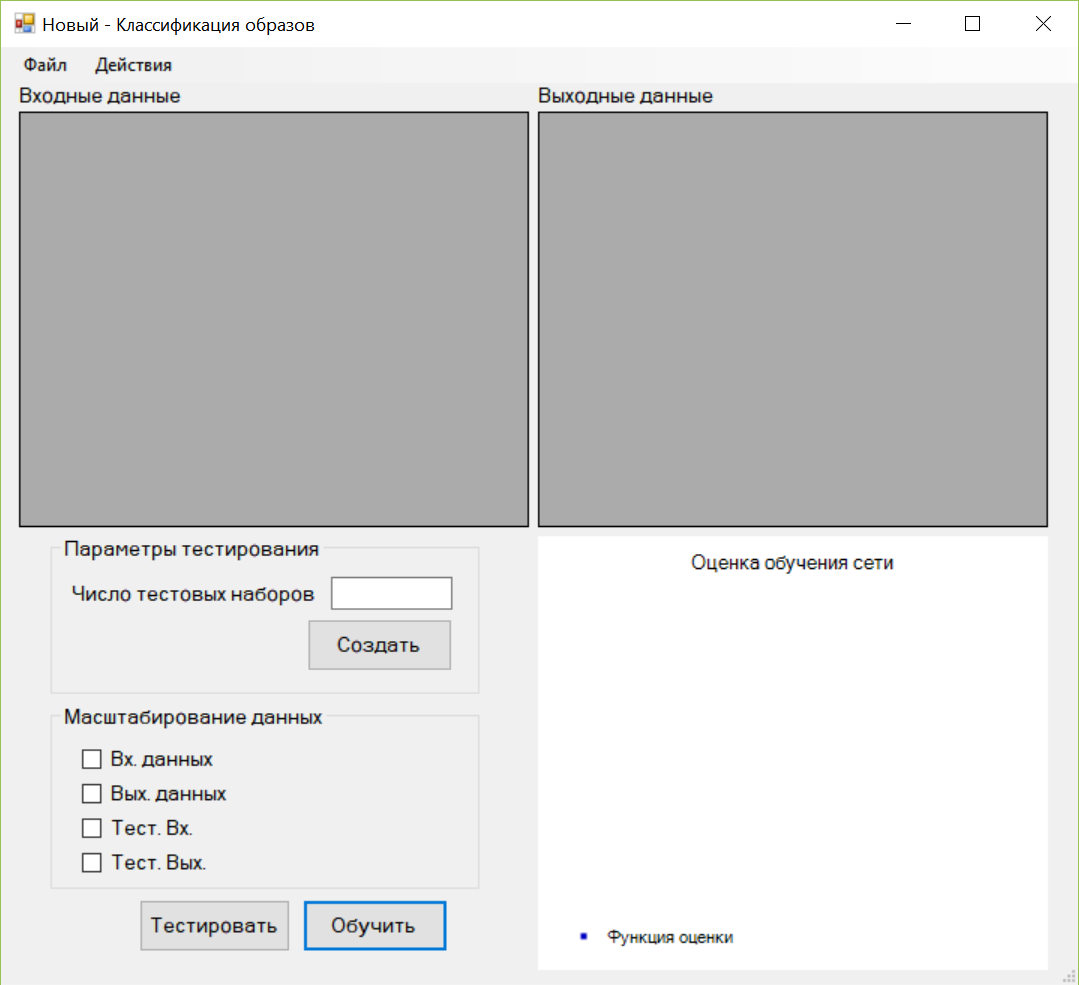


Рисунок 3.13 – Внешний вид формы «Классификация образов»

В отличие от нейронных сетей, предназначенных для решения задач регрессии, в которых, как правило, имеется много входов и один выход, в нейронных сетях для классификации присутствует много входов и много выходов. Так как число выходов может быть очень большим, то отсутствует необходимость построения графиков для каждого выхода сети, поэтому было принято решение предоставить пользователю в качестве оценки обучения график «Оценка обучения сети», по которому он может судить о правильности хода этого процесса.

3.2.2 Настройка параметров и создание новой сети

Главное окно программы приведено на рисунке 3.14. Оно состоит из рабочей области и строки меню. Рабочая область состоит из элементов ввода, в которые вводятся параметры создаваемой сети.

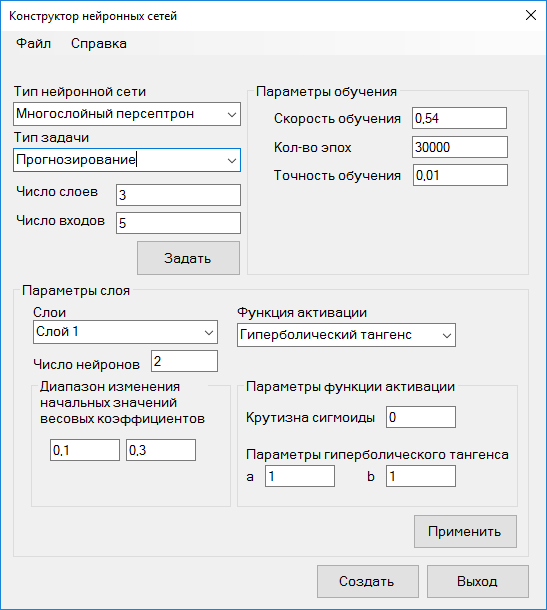


Рисунок 3.14 – Главная форма «Конструктор нейронных сетей»

В общем виде процесс создания новой сети можно представить следующим образом:

1) выбор типа сети;

2) выбрать тип решаемой задачи;

3) задать число входов и число слоёв для многослойной сети (для линейного и нелинейного нейрона задается только лишь число входов);

4) для многослойной сети указать параметры для каждого слоя (для одного нейрона данный шаг необходимо пропустить);

5) задание скорости обучения, размер обучающего множества и точность обучения.

После выполнения данных этапов запускается новая форма для решения того типа задач, который был выбран.

3.2.3 Создание одного нейрона

Линейный (нелинейный) нейрон обладает множеством входов и одним выходом. С помощью нейрона можно решать задачи регрессии и классификации.

Для создания линейного нейрона в раскрывающимся списке «Тип нейронной сети» выбрать пункт «Линейный нейрон». Далее необходимо задать число входов и нажать на кнопку «Задать». Затем необходимо указать скорость обучения. Диапазон изменения скорости обучения составляет (0;1). Если пользователь неверно указал этот параметр, на экране появится предупреждающее сообщение. Число эпох указывает на размер обучающего множества. Точность обучения является критерием остановки обучения. Следовательно, она должна быть меньше единицы и стремиться к нулю. Следует относится внимательно к данному параметру, так как при указании высокой точности обучения будет занимать более продолженный период времени.

Кроме того, для нелинейного нейрона (персептрона) имеется возможность задавать функцию активации и ее параметры. Для этого необходимо из раскрывающегося списка «Функция активации» выбрать необходимый пункт и задать соответствующие ей параметры, затем нажать на кнопку «Принять».

После того как все параметры будут указаны необходимо нажать на кнопку «Создать» или выбрать соответствующий пункт меню Файл -> Создать. На экране появится новая форма. Процесс запуска обучения во всех случаях будет примерно одинаковым. Для этого первым делом нужно подготовить входные данные. В программе имеется возможность загрузки исходных данных для обучения из файла, а также их ввод пользователем вручную в таблицы «Входные данные» и «Эталон».

Для загрузки данных необходимо выбрать соответствующий пункт из строки меню. Для загрузки обучающего набора необходимо выбрать пункт Действия-> Загрузить обучающий набор. Для загрузки эталонов необходимо выбрать пункт Действия -> Загрузить набор эталонов. После того как данные загрузятся в таблицу необходимо нажать на кнопку «Обучить» и запуститься процесс обучения. В ходе этого процесса пользователь может наблюдать ход обучения в графическом виде. На первом графике «Выход нейрона» пользователь может наблюдать реальный и желаемый выходы. На втором графике «Оценка обучения сети» приводится график целевой функции, которую необходимо минимизировать в процессе обучения. На третьем графике «Ошибка» показана разница между текущим выходом и эталоном на текущей итерации. Внизу окна можно наблюдать счетчик времени обучения. Обучение заканчивается по достижению указанной точности.

Принудительно можно остановить обучение, нажав на кнопку «Остановить». Результат работы можно сохранить, для этого в пункте меню необходимо выбрать пункт Файл -> Сохранить.

После обучения имеется возможность тестирования полученных результатов. Для этого подобным образом загружаются тестовые наборы входных данных, в поле «Число тестовых наборов» указывается число итераций тестирования. Запуск происходит по нажатию на кнопке «Тестировать». Результат выполнения операции можно увидеть на экране в виде графиков.

Кроме этого для нелинейного нейрона имеется ряд дополнительных возможностей, а именно возможность масштабирования входных и выходных данных. Для того чтобы воспользоваться данной возможностью необходимо поставить галочку в пункте «Масштабирование данных». Данный процесс полностью скрыт от пользователя, что позволяет избежать проблем с использование данного функционала.

3.2.4 Создание многослойной сети

«Конструктор нейронных сетей» позволяется создать многослойные сети для решения двух типов задач: регрессии и классификации образов.

Для создания сети для классификации образов необходимо из раскрывающегося списка «Тип нейронной сети» выбрать пункт «Многослойный персептрон», а в раскрывающимся списке «Тип задачи» выбрать пункт «Классификация образов». Для решения задач регрессии в списке «Тип задачи» необходимо выбрать пункт «Прогнозирование». Дальнейшая настройка параметров для данных типов сетей происходит подобным образом.

После указания типа нейронной сети и типа решаемой задачи необходимо указать число слоев и число входов, данные параметры вводятся в соответствующие строки ввода. Для задания параметров необходимо нажать на кнопку «Задать».

Далее необходимо задать параметры для каждого слоя сети, для этого в раскрывающимся списке «Слои» выбирается нужный слой. Имеется возможность указать число нейронов, диапазон изменения начальных значений весовых коэффициентов, а также тип активационной функции. Для активационной функции имеется возможность задать ее коэффициенты, по умолчанию эти значения равны единице. При неверном задании числа нейронов, пользователь получит уведомление.

После задания параметров слоя необходимо нажать на клавишу «Применить», чтобы применить настройки. Данную операцию следует повторить для каждого слоя.

Затем необходимо указать скорость обучения, число обучающих наборов, точность обучения и нажать на кнопку «Создать».

В зависимости от типа решаемой задачи запустится новая форма. Формы имеют схожий внешний вид за исключением некоторых отличий.

Настройки сети можно сохранить в файл или загрузить из файла, для этого нужно выбрать соответствующий пункт из пункта меню «Файл».

При запуске формы создается новая таблица для входных данных. Число столбцов таблицы равно число входов, а число строк равно числу обучающих наборов. Для таблицы эталонов, число столбцов равно число нейронов в выходном слое. Пользователь может записать входные данные в таблицу вручную, а также загрузить из файла. Для загрузки данных из файла необходимо выбрать пункт меню Действия -> Загрузить обучающий набор или Действия ->Загрузить набор эталонов. При ошибке открытия файла, по каким-либо причинам, появляется предупреждающее сообщение.

Процесс обучения начинается после нажатия на кнопку «Обучить». Обучение длится до тех пор, пока не будет достигнута заданная точность, или пока пользователь не нажмет на кнопку «Остановить». После завершения процесса обучения можно протестировать результат работы сети. Для этого необходимо загрузить обучающий набор входов и эталонов. В строке «Число тестовых наборов» необходимо указать число итераций и нажать на кнопку «Создать». Тестирования начинается после нажатия на кнопку «Тестировать». Если результаты тестирования устраиваю, то их можно сохранить в файл, для этого необходимо воспользоваться командой Файл-> Сохранить.

3.2.5 Общие рекомендации по работе с программой

Обучение нейронных сетей является очень ресурсоемким процессом, поэтому необходимо очень тщательно подходить к выбору архитектуры сети и настройки ее параметров. Кроме того, огромную роль оказывает аппаратное обеспечение машины, на которой будет запускаться «Конструктор нейронных сетей». Так, например, на многоядерных процессорах, процесс обучения будет выполняться горазда быстрее, чем на мобильных процессорах.

При выборе параметров первым делом необходимо обратить внимание на количество слоев и количество нейронов в них. Многослойная сеть является очень мощным вычислительным механизмом, поэтому для большинства задач требуется порядка десяти нейронов в каждом слое. Не рекомендуется создавать более 3 слоев, так как это существенно замедляет обучение. Для таких сетей существуют более продвинутые методы глубинного обучения. Важно обратить внимание на скорость обучения диапазон которой изменяется (0;1). При высоких значениях этого параметра, например, 0.9, скорость обучения будет достаточно высокой, однако существует возможность попасть в локальный минимум и на этап этапе произойдет паралич сети. Фактически желаемая точность не будет достигнута. При слишком малых значениях скорость обучения будет заметно снижаться, поэтому рекомендуется начинать с высоких значений скорости обучения и снижать ее, если это будет действительно необходимо.

Более подробно следует рассмотреть процесс выбора количества нейронов в слоях сети. Как уже говорилось ранее, их число не должно быть слишком высоким, так как это замедляется процесс обучения. Кроме этого может возникнуть, так называемое, переобучение сети. Данная проблема возникает в случае, если сеть обладает чрезмерными вычислительными ресурсами, таким образом, она обучается точно повторять поведение, которое советовало входным обучающим данным, однако при подаче на вход переобученной сети могут происходить всплески, что говорит о не правильной интерпретации выходного сигнала.

Так как для обучения сети необходимо последовательно подавать на вход обучающие данные и сравнивать реальный выход сети с его желаемым аналогом, поэтому необходимо обратить внимание на некоторые правила по выбору обучающих данных. Первым делом необходимо обратить внимание на то, что для того, чтобы как можно лучше обучить сеть нужно обладать очень большим набором данных. Чем больше таких данных, тем лучших результатов можно добиться в процессе обучений. Следующим на что следует обратить внимание — это то, что выход активационной функции как правило расположен от [0;1] или от [-1;1]. Поэтому необходимо принять меры для подготовки входных и выходных данных. Разработанная программа позволяет шкалировать входные и выходные данные, используяSoftmax-шкалирование.

**3.3.1 Назначение и условия применения программы**

**Программное средство «Конструктор нейронных сетей» предназначено для конструирования и тестирования различных типов многослойных сетей. При разработке данного средства была составлена библиотека классов Neuron.dll. Данную библиотеку можно применять для разработки приложений специфической направленности, в состав которых входят многослойные нейронные сети. При разработке программного обеспечения следует учитывать, что минимальные требования к платформе – это .NET Framework 4.5. Это связано с тем, что в методах используются средства, обеспечивающие параллельное выполнения задач, которые призваны увеличить скорость работы реализованных алгоритмов.**

**3.3.2 Входные и выходные данные**

**3.3.3 Назначение, состав и применение программных блоков**

**Программное обеспечение для построения нейронных сетей «Конструктор нейронных сетей» состоит из:**

**- исполняемого файла КонструкторНейросетей.exe;**

**- библиотеки классов Neuron.dll.**

**Главным классом является класс MainForm. При запуске программы создается объект данного класса и на экране пользователя отображается главная форма. Данная форма является интерфейсом для ввода пользователем параметров.**

**Ключевым является класс AppSettings, который предназначен для хранения настроек и обмена данными между формами. Данный класс реализует паттерн «Одиночка», т. е. можно создать только один экземпляр данного класса.**

**Классы NeuralNetForm, LinearNeuronForm, NonLinearNeuronForm, ClassificationForm является классами для создания формы, на которой расположены различные элементы графического пользовательского интерфейса. При создании объекта данных классов, в качестве параметра конструкта передается ссылка на объект класса AppSettings.**

Непосредственно ядром программы можно назвать библиотеку классов «Neuron.dll». Данная библиотека содержит набор классов необходимых для создания нейронных сетей, которые, в свою очередь, содержат необходимые для этого методы и свойства.

**Для того чтобы воспользоваться библиотекой классов, которая содержит классы непосредственно для создания нейронных сетей, необходимо добавить в проекте ссылку на эту библиотеку.**

3.4 Результаты экспериментальных исследований

В связи с тем, что обучение нейронных сетей является весьма долгим процессом и требует огромного количества тестовых данных. Причем для решения конкретной задачи требуются специфичные данные, что может кардинально влиять на выбор архитектуры сети и ее параметров.

Поэтому было принято решение создать тестовый набор данных модуля функции синуса в количестве 30000 отсчетов и на этом примере продемонстрировать влияние различных параметров на скорость обучения. Все сети имеют 3 слоя. Диапазон изменения весовых коэффициентов расположен в диапазоне от 0.3 до 0.5. Параметры активационной функции равны 1. Сеть имеет 5 входов и один выход. Результаты тестирования приведены в таблице 3.1.

На рисунке 3.22 покан результат обучения первой нейронной сети, имеющей по 5 нейронов в скрытых слоя и сигмоидальную функцию активации. Скорость обучения равна 0.9каких единиц измерения.

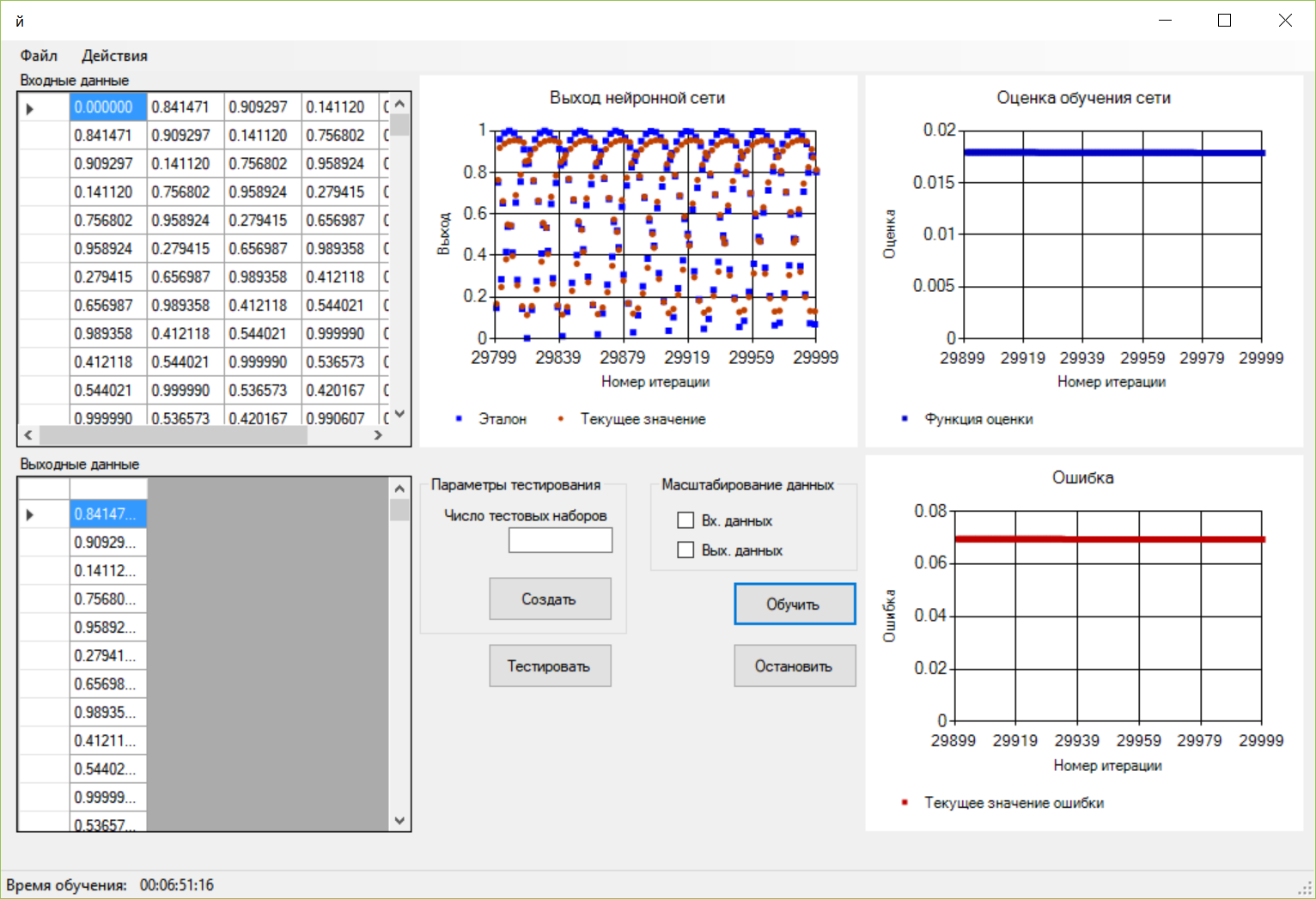


Рисунок 3.22– Результат обучения нейронной сети №1

Из рисунка видно, что значения выходов сети близко расположены к эталонным значениям. Это говорит о том, что сеть действительно пытается обучиться предсказывать значения функции. Для получения более точных значений, вероятно, придется либо изменить число нейронов, скорость обучения, либо повторить обучение на текущих примерах до получения значений выходов сети с заданной точностью.

На рисунке 3.23 приведен результат обучения второй нейронной сети. В данном случае активационная функция для скрытых слоев изменена на гиперболический тангенс.

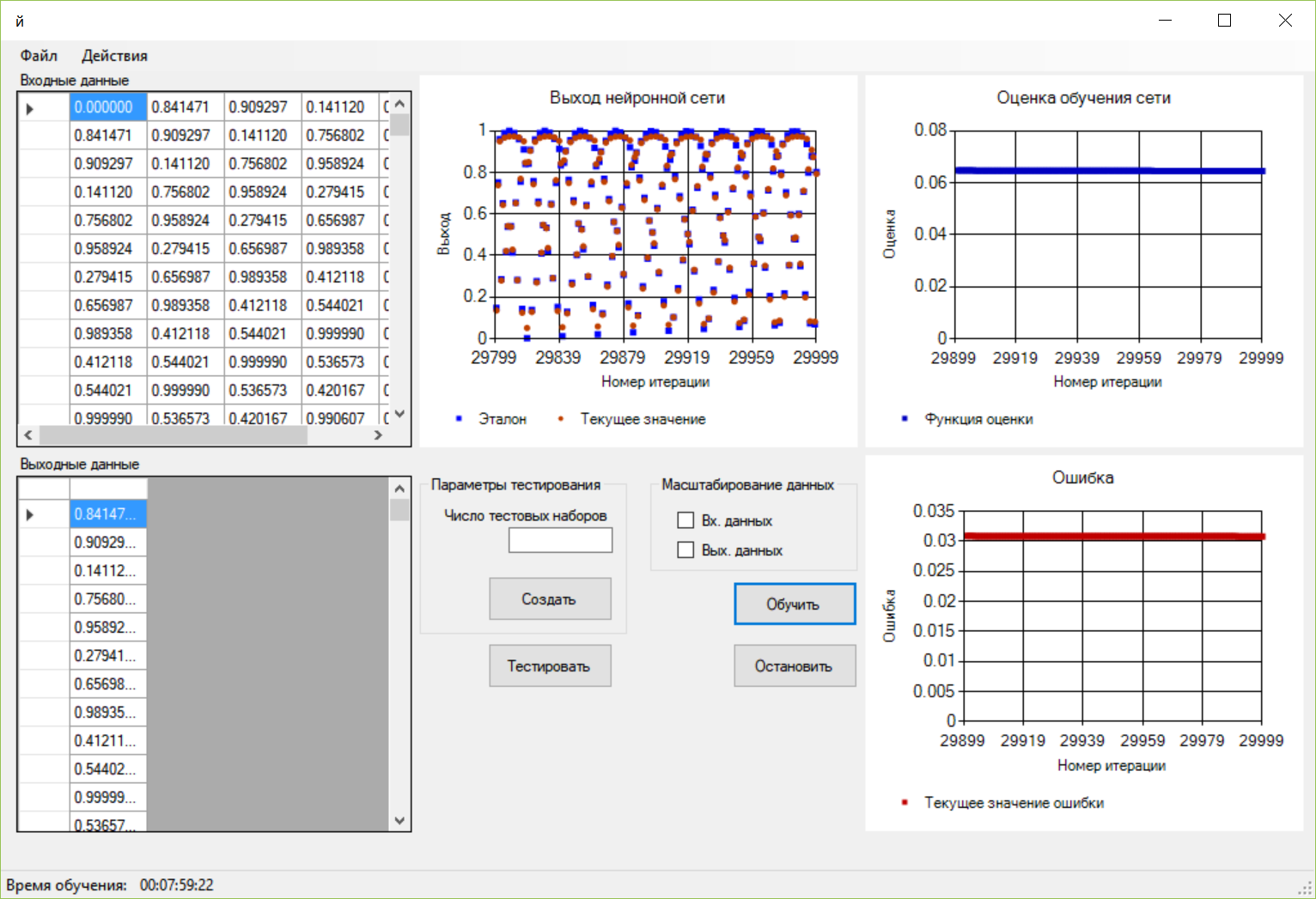


Рисунок 3.23 – Результат обучения нейронной сети №2

Сравнив результаты, показанные на рисунке 3.22 и 3.23 видно, что на втором рисунке результатом обучения являются выходы сети, имеющие более высокую точность, чем на первом рисунке. Если сравнивать значения целевой функции, то тут во втором случае значение целевой функции будет выше, чем в первом, но при более высокой точности. Поэтому можно предположить, что во втором случае обучение идет медленнее, это связано с тем, что для вычисления значения гиперболического тангенса требуется выполнить больше арифметических операций. Однако обучение происходит точнее.

На рисунке 3.24 показан результат обучения третьей нейронной сети. В данном случае скорость обучения была снижена до значения 0.1. Активационной функцией для всех слоев является сигмоидальная функция.

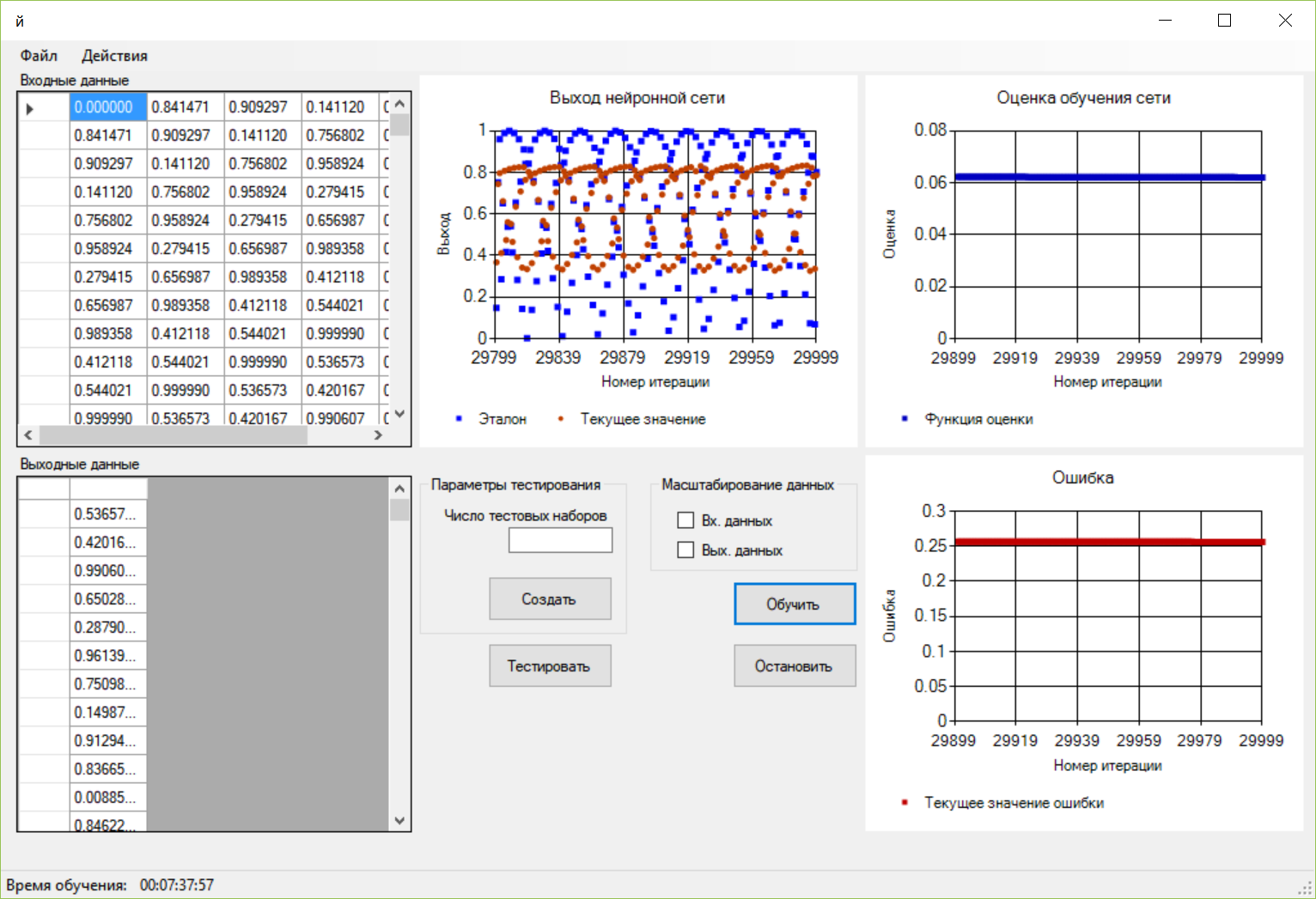


Рисунок 3.24 – Результат обучения нейронной сети №3

Из рисунка можно увидеть, что выходы сети обладают более высокой ошибкой по отношению к целевым выходам в отличие от прошлого опыта. Можно сделать вывод, что при уменьшении скорости обучения, время обучения увеличивается. Порой данное действие является необходимым, например, когда целевая функция достигает локального минимума. И чтобы выйти из него, одним из вариантов является снижение скорости обучения.

На рисунке 3.25 приведен результат обучения для нейронной сети подобной третьему номеру опыта, разница заключается в типах активационной функции в скрытых слоях. В данном случае в них расположена функция гиперболического тангенса.

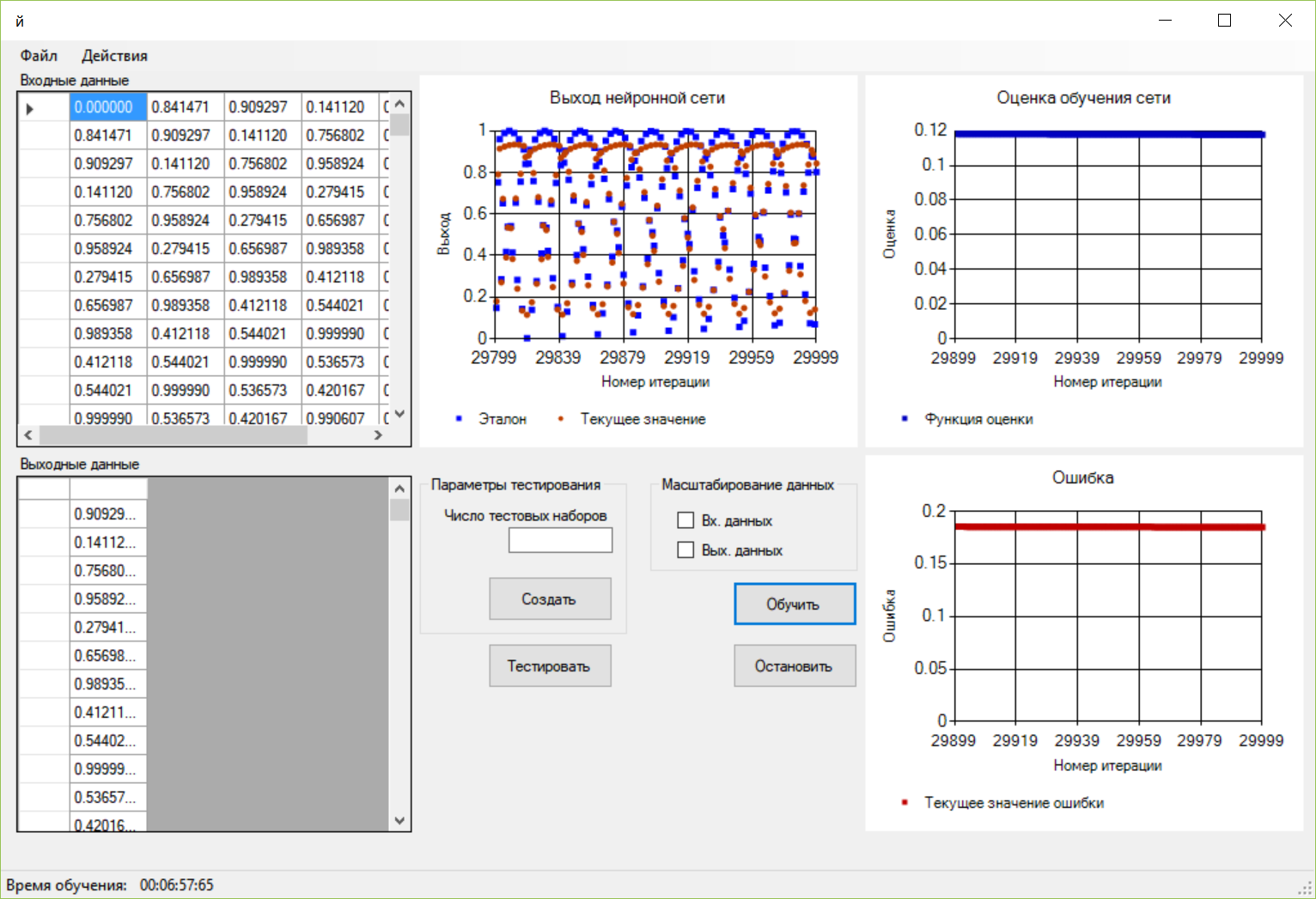


Рисунок 3. 25 - Результат обучения нейронной сети №4

Как и следовало ожидать из предыдущих опытов, наличие разных типов активационных функций повышает обучаемость сети. Однако для достижения требуемой точности, при малых значениях скорости обучения требуется большее количество итераций обучения.

В следующих примерах число нейронов в скрытых слоях было увеличено до 50. Ошибка обучения для последующих опытов является одинаковой. Результат обучения приведен на рисунке 3.26.

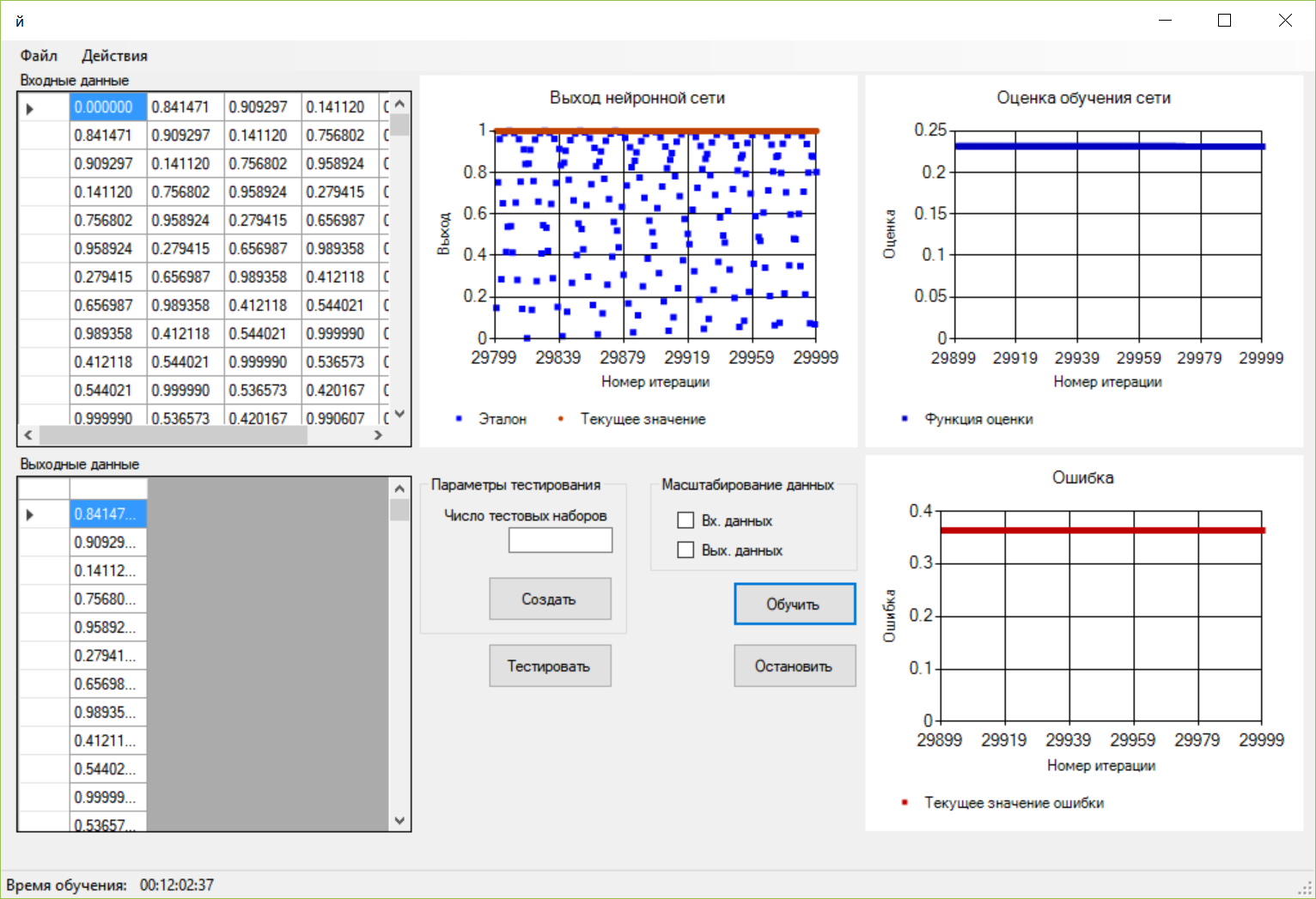


Рисунок 3.26 - Результат обучения нейронной сети №5 – 8

Из рисунка 3.26 видно, что выход сети расположен около единицы, это вызвано тем, что на вход нейрона подаются значения около или равные единицы. Как и следовало ожидать, увеличение число нейронов заметно снижают скорость обучения. В данном примере, такое количество нейронов является избыточным. Высока вероятность, что сеть переобучится, если подавать ей обучающий набор до тех пор, пока, не будет достигнута точность.

Таблица 3. 1 – Результаты работы программы «Конструктор нейросетей»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Тип активационной функции | Число нейронов в слое | Скорость обучения | Время обучения | Среднеквадратическая ошибка | Текущая ошибка |
| 1 | Сигмоидальная – сигмоидальная – сигмоидальная | 5 - 5 -1 | 0.9 | 6 мин. 51 сек. | 0.017 | 0.069 |
| 2 | Гиперболический тангенс – гиперболический тангенс – сигмоидальная | 5 - 5 -1 | 0.9 | 7 мин. 59 сек. | 0.064 | 0.031 |
| 3 | Сигмоидальная – сигмоидальная – сигмоидальная | 5 - 5 -1 | 0.1 | 7 мин. 37 сек. | 0.062 | 0.256 |
| 4 | Гиперболический тангенс – гиперболический тангенс – сигмоидальная | 5 - 5 -1 | 0.1 | 6 мин. 57 сек. | 0.117 | 0.184 |
| 5 | Сигмоидальная – сигмоидальная – сигмоидальная | 50 - 50 -1 | 0.9 | 12 мин. 2 сек. | 0.231 | 0.363 |
| 6 | Гиперболический тангенс – гиперболический тангенс – сигмоидальная | 50 - 50 -1 | 0.9 | 11 мин. 44 сек. | 0.344 | 0.363 |
| 7 | Сигмоидальная – сигмоидальная – сигмоидальная | 50 - 50 -1 | 0.1 | 11 мин. 43 сек. | 0.457 | 0.363 |
| 8 | Гиперболический тангенс – гиперболический тангенс – сигмоидальная | 50 - 50 -1 | 0.1 | 16 мин. 19 сек. | 0.571 | 0.363 |

Проанализировав результаты из таблицы 3.1 можно сделать вывод, что число нейронов в слоях заметно влияют на время обучения. Соответственно, чем больше нейронов, тем больше вычислительная мощность сети и тем большее время требуется на ее обучение. Уменьшение параметра скорости обучения также заметно увеличивает время обучения. Однако уменьшение этого параметра порой оправдано, так как это позволяет обходить локальные минимумы целевой функции. Комбинирование различных типов активационных функций порой улучшает качество обучения, не влияя на его скорость, вероятно, это связано с тем, что каждая функция вносит больше возмущения, что и влияет на качество обучения.

При построении нейронных сетей нужно очень аккуратно подходить к выбору архитектуры и настройке параметров. Начинать стоит с небольшого количества нейронов и достаточно высокой скорости обучения. Затем стоит обратить внимание, как ведет себя целевая функция. Если ее значение стремительно уменьшается, значит, архитектура выбрана правильно. Если же функция активации колеблется между некоторым диапазоном значений и требуемая точность не достигнута, то следует, либо увеличить число нейронов в слоях сети, либо уменьшать скорость обучения, чтобы выбраться из локального минимума.

Разработанное приложение позволит пользователю решать огромный спектр практических задач. К ним можно отнести задачи регрессии и классификации. Кроме того, приложение будет полезно для тех, кто только начинает изучать нейронные сети, так и для тех, кто обладает большим набором данных, для которых требуется построение нейросетевой модели.

СПИСОК ЛИТЕРАТУРЫ

1 Хайкин С. Нейронные сети: полный курс /С. Хайкин. - 2-е изд., испр.- пер. с англ. д. т. н. Н. Н. Куссуль. - М: ООО «И.Д. Вильямс», 2006. – 1104 с.

2Ясницкий Л. Н. Введение в искусственный интеллект / Л. Н. Ясницкий. – М.: Издательский центр «Академия», 2008. – 176 с.

3Джиган В. И. Адаптивная фильтрация сигналов: теория и алгоритмы. /В. И. Джиган. - М.: Техносфера, 2013. – 528 с.

4Каллан Р. Основные концепции нейронных сетей. / Р. Каллан. - пер. с англ. А.Г. Сивака. - М: ООО «И.Д. Вильямс», 2001. – 287 с.

5 Круглов В. В. Искусственные нейронные сети. Теория и практика, 2-е изд., стереотип. / В. В. Круглов, В. В. Борисов. – М: Горячая линия-Телеком, 2002. – 382 с.

6 Медведев В. С. Нейронные сети. MATLAB 6 / В. С. Медведев, В. Г. Потемкин: под общ. ред. к. т. н. В. Г. Потемкина. – М: ДИАЛОГ-МИФИ, 2002. – 496 с.

7 Ясницкий Л. Н. Искусственный интеллект. Элективный курс: Учебное пособие / Л. Н. Ясницкий. – М.: БИНОМ. Лаборатория знаний, 2011. – 240 с.

8 Круг П. Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу «Микропроцессоры» / П. Г. Круг. – М.: Издательство МЭИ, 2002. – 176 с.

9 Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Э. Троелсен – 6-е изд. – пер. с англ. Ю. Н. Артеменко. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.