

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по выполнению практических работ

по дисциплине

«ИСКУССТВЕННЫЕ НЕЙРОННЫЕ СЕТИ»

для студентов направления подготовки 02.04.01 Математика и компьютерные
науки

профиль «Искусственный интеллект в кибербезопасности»

**Ставрополь
2026**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ	5
ПРАКТИЧЕСКАЯ РАБОТА 1. Основы работы с программной средой компьютерной математики	5
ПРАКТИЧЕСКАЯ РАБОТА 2. Пакет моделирования нейронных сетей neural network toolbox	30
ПРАКТИЧЕСКАЯ РАБОТА 3. Пакет моделирования нейронных сетей Neural Fitting	42
ПРАКТИЧЕСКАЯ РАБОТА 4. Изучение свойств линейного нейрона и линейной нейронной сети	51
ПРАКТИЧЕСКАЯ РАБОТА 5. Изучение многослойного нелинейного персептрона и алгоритма обратного распространения ошибки	69
ПРАКТИЧЕСКАЯ РАБОТА 6. Применение нейронных сетей для распознавания образов	90
ПРАКТИЧЕСКАЯ РАБОТА 7. Разработка нейросетевого алгоритма классификации цветов ириса	96
ПРАКТИЧЕСКАЯ РАБОТА 8. Распознавание букв	105
ПРАКТИЧЕСКАЯ РАБОТА 9. Использование нейронной сети для приближения функций	116
1.	126
2.	128
3.	129
4.	129
5. Порядок выполнения самостоятельной работы студентом	130
5.1. Методические рекомендации по работе с учебной литературой	130
5.2. Методические рекомендации по составлению конспекта лекций:	131
5.3. Методические рекомендации по подготовке к практическим занятиям	132
5.4. Методические рекомендации по самопроверке знаний	133
6.	138

ВВЕДЕНИЕ

Цель и задачи освоения дисциплины

Целью освоения дисциплины «Искусственные нейронные сети» является формирование у будущего специалиста по направлению подготовки 02.04.01 Математика и компьютерные науки профиль «Искусственный интеллект в кибербезопасности» формирование и развитие набора общепрофессиональных и профессиональных компетенций.

Задачи дисциплины:

- Основные задачами дисциплины:
- формирование знаний по основам теории нейронных сетей (НС). - обучение методам и алгоритмам представления и обработки данных в нейронных сетях.
- привитие практических навыков по использованию инструментальных средств компьютерного моделирования создания, обучения и применения НС.

Перечень осваиваемых компетенций:

Код	Формулировка
ОПК-2	Способен создавать и исследовать новые математические модели в естественных науках, совершенствовать и разрабатывать концепции, теории и методы

Перечень планируемых результатов обучения по дисциплине, соотнесённых с планируемыми результатами освоения образовательной программы высшего образования

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине, характеризующие этапы формирования компетенций, индикаторов
ОПК-2 Способен создавать и исследовать новые математические модели в естественных науках, совершенствовать и разрабатывать концепции, теории и методы	ИД-1 осуществляет поиск системного программного обеспечения, понимает структуру операционной системы, предназначение основных параметров и компонентов операционной системы; использует методы инсталляции системного программного обеспечения, а также режимы его взаимодействия с операционной системой; выбирает методы	Знает состав, порядок установки и настройки системного и прикладного программного обеспечения, режимы его взаимодействия с операционной системой, умет настраивать операционную систему и выбирать отказоустойчивую конфигурацию компьютерных систем

	мониторинга компонентов операционной системы, а также режимы отказоустойчивой конфигурации	
ОПК-2 Способен создавать и исследовать новые математические модели в естественных науках, совершенствовать и разрабатывать концепции, теории и методы	ИД-2 Устанавливает и удаляет операционную систему семейства Windows, развертывает службу каталогов Windows, а также настраивает роли сервера; устанавливает и удаляет системное программное обеспечение, запускает, останавливает и настраивает службы Windows, а также использует программный маршрутизатор на базе ОС Windows; диагностирует причины сбоев работоспособности операционной системы Windows с помощью системного программного обеспечения	Оценивает все режимы функционирования сетей и систем передачи информации с помощью аппаратнопрограммных средств

СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

ПРАКТИЧЕСКАЯ РАБОТА 1. Основы работы с программной средой компьютерной математики

Цель занятия:

Освоить основные приемы работы с программной средой компьютерной математики MATLAB.

Порядок проведения занятия

1. Запуск программы.
2. Интерфейс пользователя системы MATLAB.
3. Простейшие математические вычисления.

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ 1. Запуск программы

Для запуска программы из меню ПРОГРАММЫ необходимо выбрать MATLAB (рисунок 1).

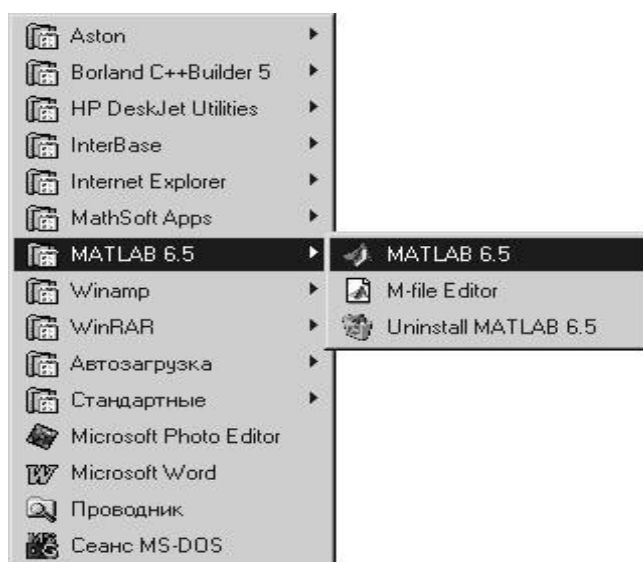


Рисунок 1 – Подготовка к запуску программы MATLAB

После инициализации блоков программы появляется основное окно программы, показанное на рисунке 2. Обычно это окно не раскрыто полностью, а занимает лишь часть экрана.

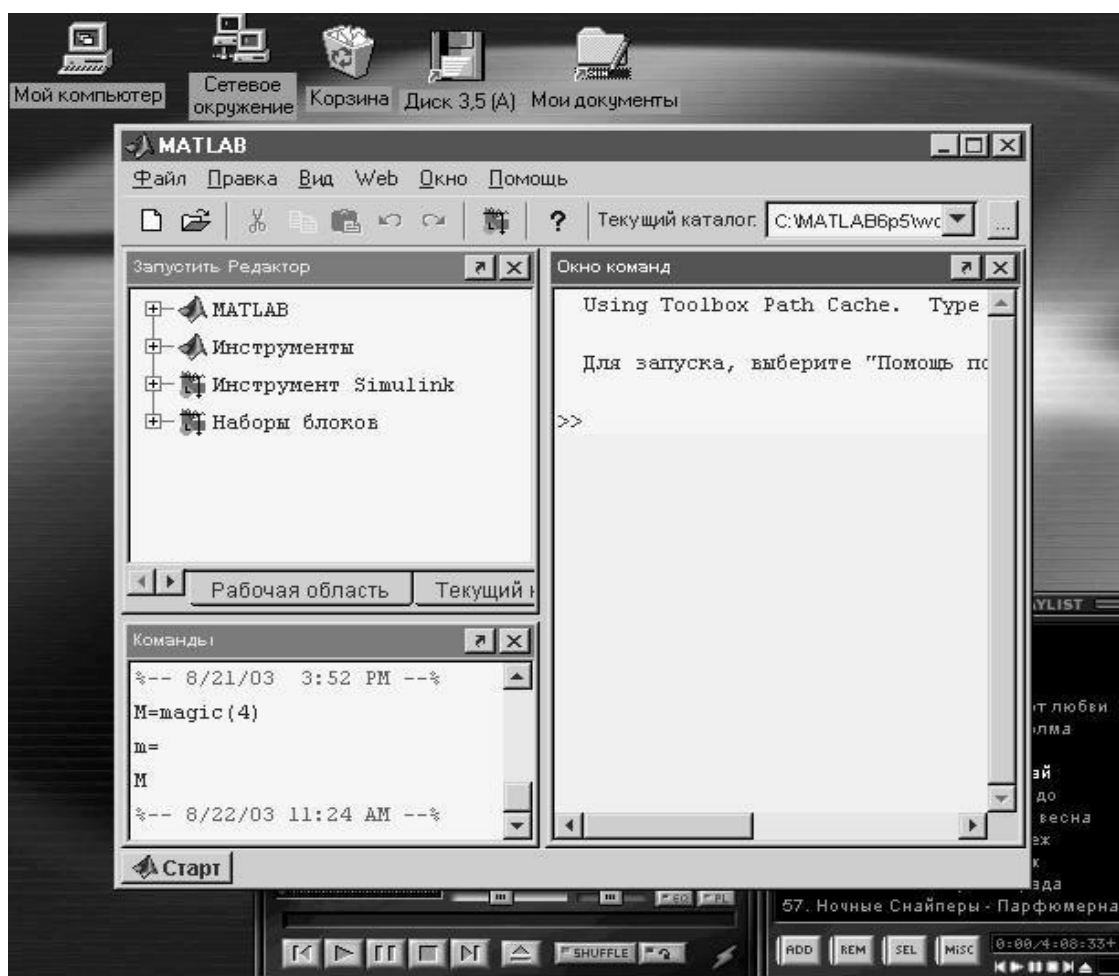
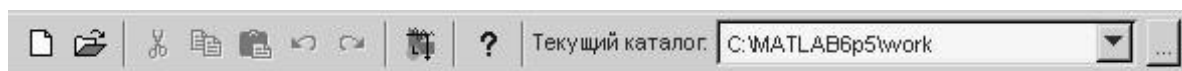


Рисунок 2 – Окно программы MATLAB после инициализации

2. Интерфейс пользователя системы MATLAB

Панель инструментов (рисунок 3) дает наиболее простой и удобный (особенно для начинающих пользователей) способ работы с системой MATLAB. При этом основные команды вводятся указанием курсором мыши на нужную кнопку с нажатием левой клавиши мыши. Кнопки имеют изображение, явно подсказывающее их назначение.



Рис

Рисунок 3 – Меню панели инструментов

Прежде всего перечислим назначение всех кнопок панели инструментов:

- New M-file (Новый m-файл) – выводит пустое окно редактора m-файлов;
- Open file (Открыть файл) – открывает окно для загрузки m-файла;
- Cut (Вырезать) – вырезает выделенный фрагмент и помещает его в буфер;

- Copy (Копировать) – копирует выделенный фрагмент в буфер;
- Paste (Вставить) – переносит фрагмент из буфера в текущую строку ввода; ☐ Undo (Отменить) – отменяет предшествующую операцию;
- Redo (Повторить) – восстанавливает последнюю отмененную операцию; ☐ Simulink – открывает окно браузера библиотек Simulink; ☐ Help (Помощь) – открывает окно справки.

Набор кнопок панели инструментов обеспечивает выполнение наиболее часто необходимых команд и вполне достаточен для повседневной работы с системой. О назначении кнопок говорят и всплывающие подсказки, появляющиеся, когда курсор мыши устанавливается на соответствующую кнопку. Они имеют вид желтого прямоугольника с текстом короткой справки. Пример такой подсказки показан на рисунке 3, у кнопки Help панели инструментов. Любопытно отметить, что доступ к браузерам рабочей области и файловой системы из панели инструментов убран.

Вызов окна открытия нового файла

Кнопка New M-file открывает окно редактора/отладчика m-файлов. Это окно показано на рисунке 4.

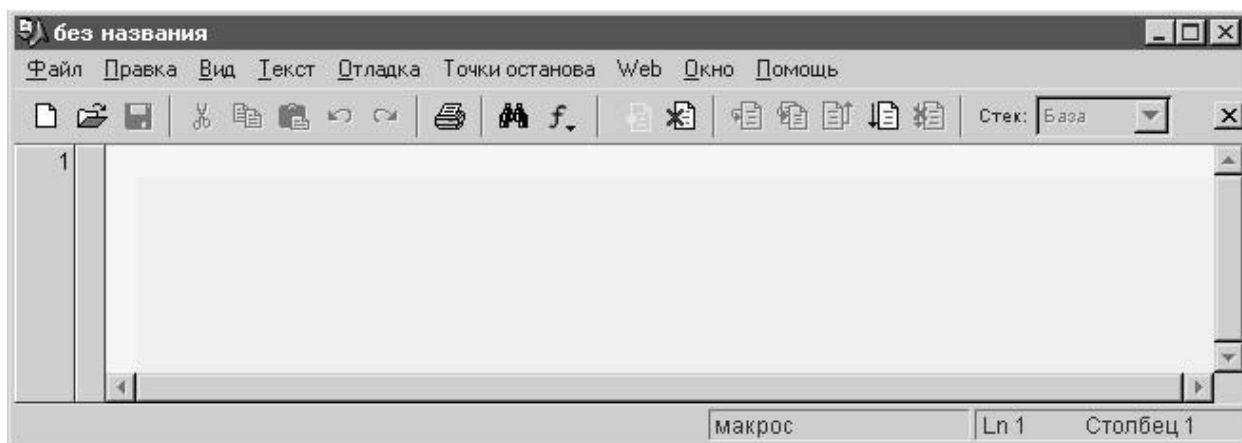


Рисунок 4 – Окно редактора/отладчика m-файлов

По умолчанию файлу дается имя без названия, которое впоследствии (при записи файла) можно изменить на другое, отражающее тему задачи. Это имя отображается в титульной строке окна редактирования m-файла, которое размещается в окне редактора/отладчика и видно на рисунке 4. В редакторе/отладчике можно редактировать несколько m-файлов, и каждый из них будет находиться в своем окне редактирования, хотя активным может быть только одно окно, расположенное поверх других окон.

Вызов окна загрузки имеющегося файла

Кнопка Open file (Открыть файл) служит для загрузки в редактор/отладчик ранее созданных m-файлов, например входящих в пакет расширения (Toolbox) системы или разработанных пользователем. Она открывает окно, которое является типичным элементом интерфейса Windows-приложений и показано на рисунке 5 внутри окна редактора/отладчика m-файлов.

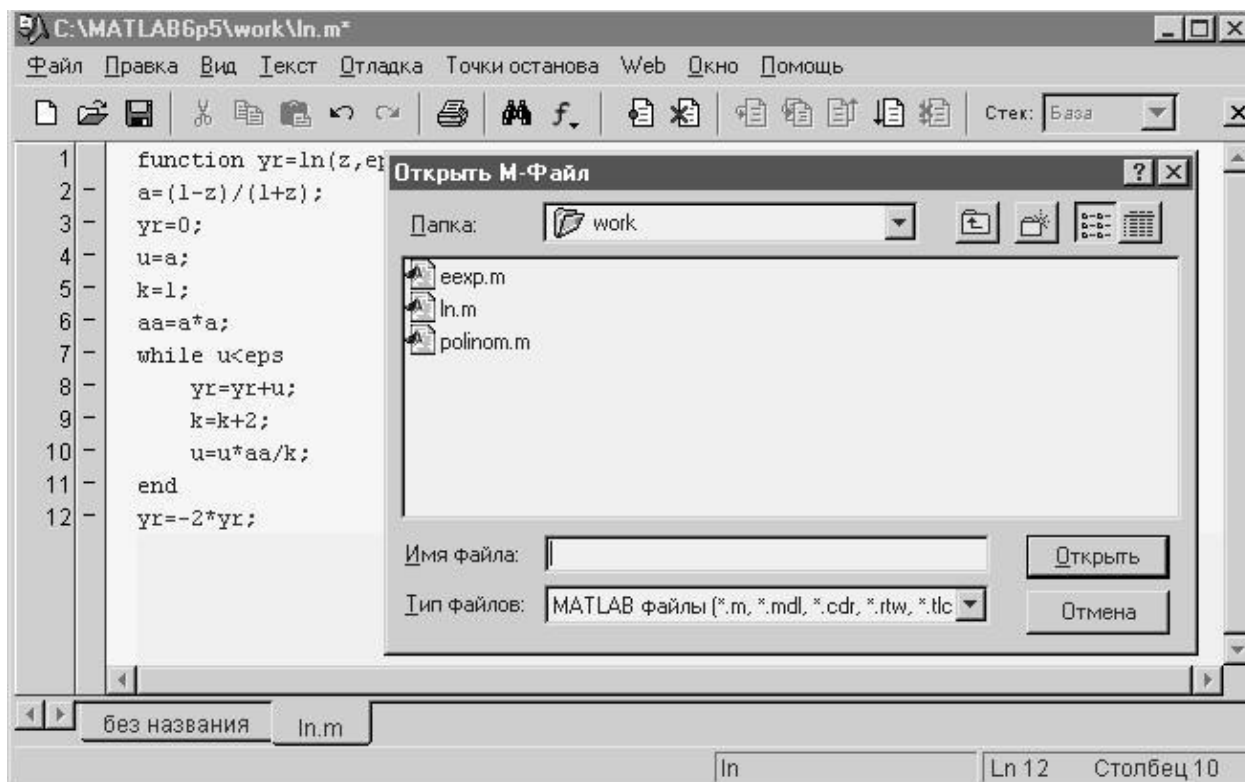


Рисунок 5 – Окно загрузки файла в окне редактора/отладчика

В окне загрузки файлов с помощью раскрывающегося списка Папка и вертикальной полосы прокрутки можно «пройтись» по всем дискам, папкам и файлам. Чтобы выбрать нужный файл для загрузки, его требуется выделить мышью. Выбор завершается двойным щелчком на имени файла, нажатием клавиши Enter или щелчком на кнопке Открыть. Любое из этих действий приводит к загрузке документа в текущее окно системы. Кнопка Отмена или клавиша Esc позволяет отказаться от загрузки.

Для вызова одного из ранее использовавшихся документов достаточно щелкнуть мышью на его имени в списке, находящемся над командой Выход в меню Файл. После загрузки файла с документом его текст появляется в текущем. Его можно начинать редактировать или запускать на исполнение.

Операции с буфером обмена

Кнопки Вырезать, Копировать и Вставить реализуют наиболее характерные команды работы с буфером обмена (Clipboard). Первые две операции относятся к выделенным фрагментам сессии или текста m-файлов (если они выполняются в окне редактора/отладчика). Для выделения объектов можно использовать мышь, перемещая курсор по тексту при нажатой левой кнопке, или клавиши со стрелками в комбинации с клавишей

Shift.

На рисунке 6 показан пример выделения содержимого матрицы M в окне документа MATLAB. Эта матрица формируется функцией `magic(n)` и называется магической, поскольку сумма элементов любого столбца, любой строки и даже любой диагонали равна одному и тому же числу – 34 для матрицы при $n=4$.

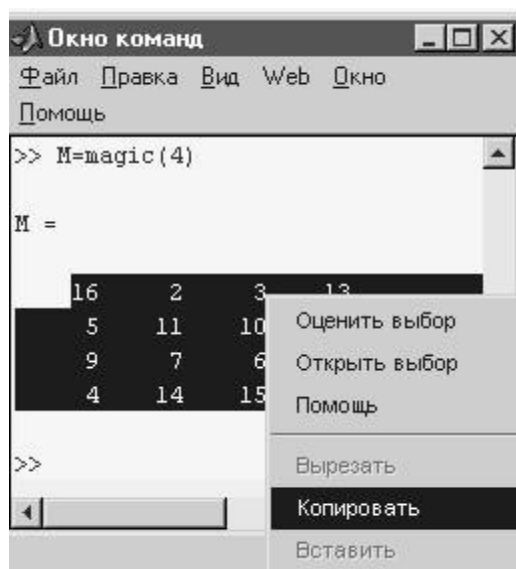


Рисунок 6 – Окно документа с выделенным содержимым матрицы M

Команда Cut (Вырезать) осуществляет вырезание выделенного фрагмента и размещение его в буфере. При этом вырезанный фрагмент удаляется из текста документа. Команда Копировать просто копирует выделенный фрагмент в буфер, сохраняя его в тексте. Команда Вставить вызывает объект из буфера (сохраняя объект в буфере) и помещает копию объекта на место в документе, указанное текстовым курсором. Эти операции реализуются как соответствующими кнопками, так и командами меню Редактировать.

В MATLAB можно использовать контекстное меню, появляющееся при нажатии правой кнопки мыши. Например, установив курсор мыши на выделенный фрагмент матрицы M и нажав правую кнопку, можно увидеть меню, показанное на рисунке 6. В нем, кстати, дублируется позиция с командой Сору (Копировать). Есть и ряд других доступных в данный момент команд. Обратите внимание, что в момент подготовки магической матрицы M ее имя появилось в окне браузера рабочей области — в правой части экрана. При этом матрица представляется изображением таблицы.

Содержимое буфера можно перенести в строку ввода, в окно редактора-отладчика m-файлов или даже в другое приложение. Допустим, мы хотим создать матрицу M1 с содержимым, которое размещено в буфере. Для этого достаточно набрать M1=[и нажав правую клавишу мыши, выбрав из появляющегося меню команду Вставить (рисунок 7).

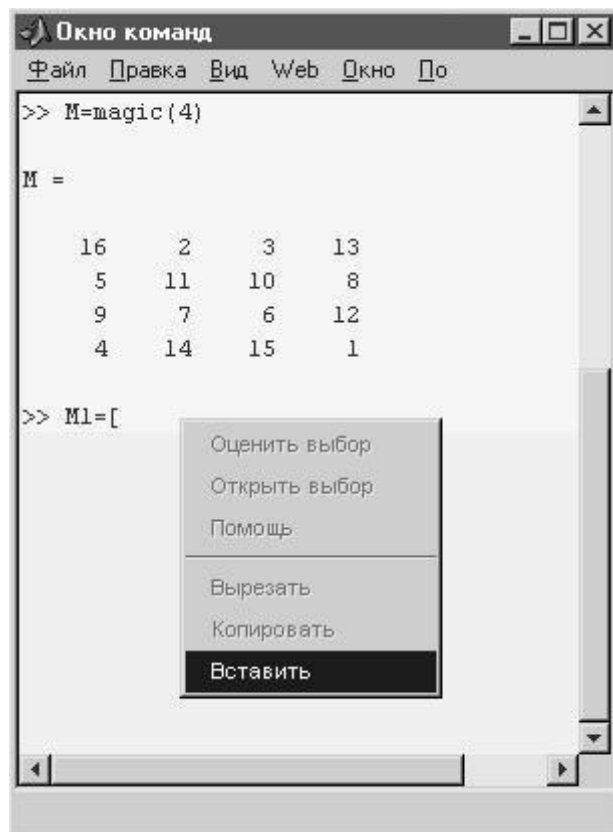


Рисунок 7 – Подготовка к вставке данных матрицы из буфера

Исполнив команду Paste, можно увидеть, что данные хранящейся в буфере матрицы появятся после открывающей квадратной скобки. Для создания матрицы M1 остается ввести закрывающую квадратную скобку] и нажать клавишу Enter. На рисунок 8 показано, как создается матрица M1, по содержимому аналогичная матрице M.

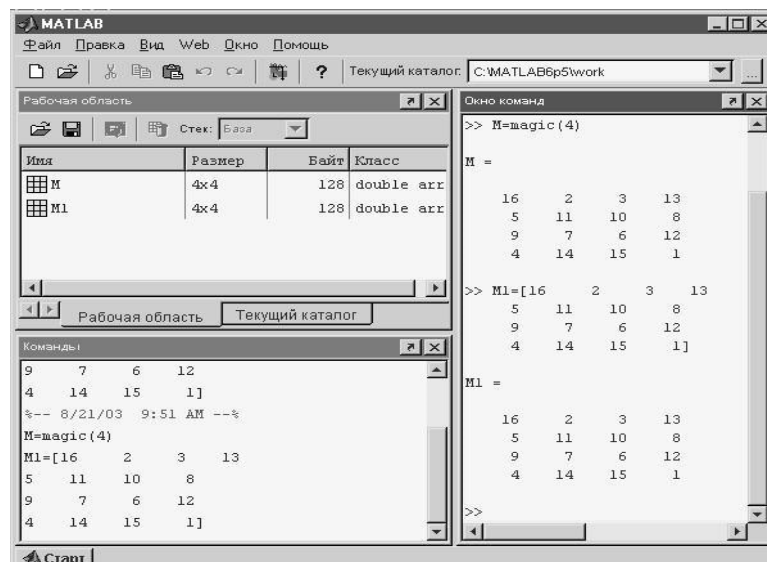


Рисунок 8 – Пример создания матрицы M1 с содержимым, взятым из буфера

Разумеется, этот пример является чисто учебным. Не обращаясь к помощи буфера, можно было бы просто записать M1=M. Все примеры в тексте методической разработки получены переносом выделенных фрагментов соответствующей сессии в окно текстового

редактора Microsoft Word. Возможен и обратный перенос – записанных в документах редактора Microsoft Word примеров в командную строку MATLAB для исполнения примеров.

Отмена результата предшествующей операции

Часто, выполнив какую-то операцию, мы отмечаем, что она оказалась ошибочной. При работе в MATLAB такой ситуации пугаться не стоит, — нажатие кнопки Undo (Отменить) панели инструментов приведет к отмене последнего действия, выполненного в текущей строке. Операции в предыдущих строках документа этой командой не отменяются. Если оказалось, что вы зря произвели отмену последней операции, то ее легко восстановить, введя с панели инструментов операцию Redo (Восстановить).

Вызов справки MATLAB

Последняя кнопка панели инструментов Help (Помощь) открывает окно с перечнем разделов справочной системы. Это окно было показано на рисунке 9.

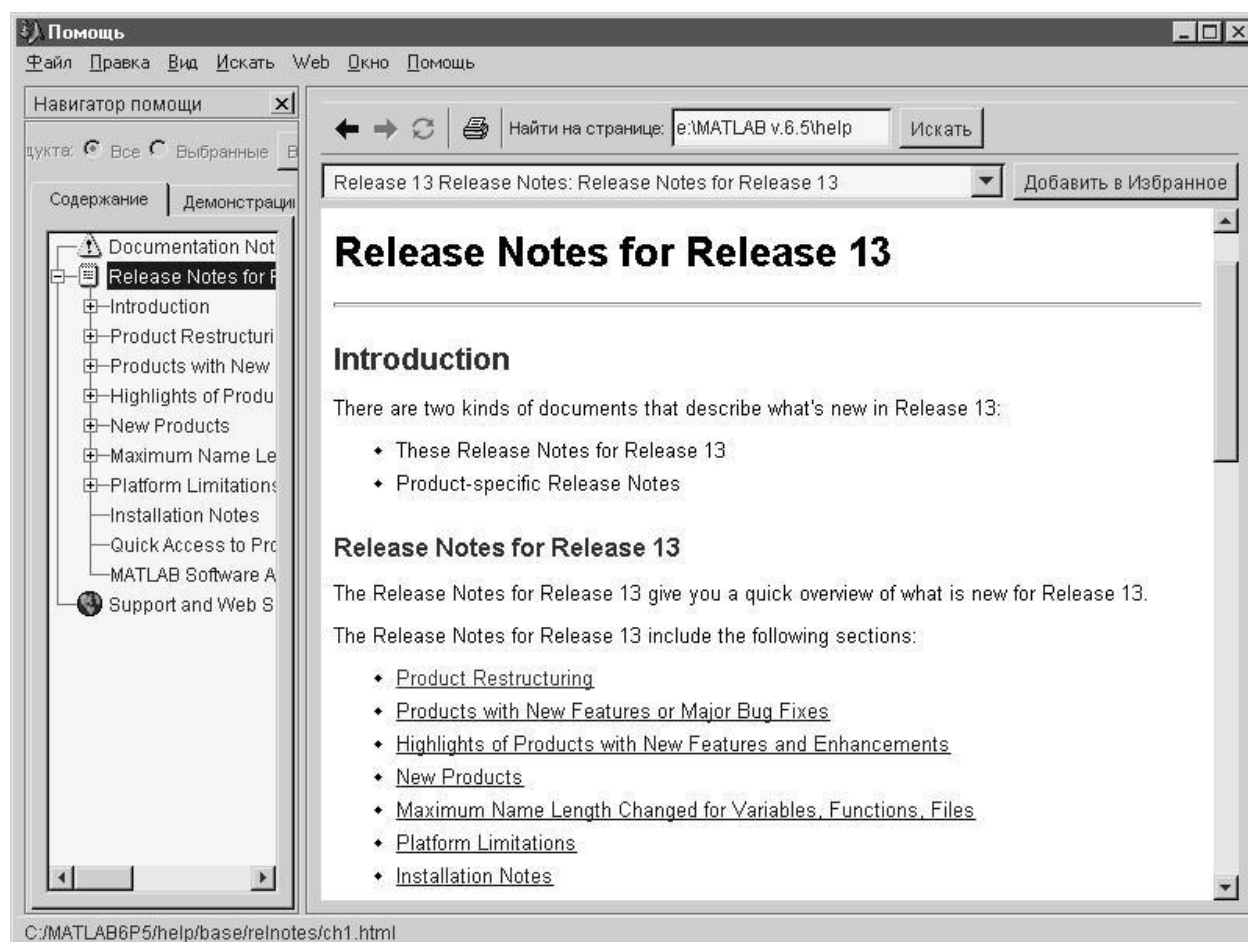


Рисунок 9 – Помощь по MATLAB

Средства контроля рабочей области и файловой системы Браузер рабочей области

Векторы и матрицы могут занимать большой объем памяти. Конечно, речь не идет о векторах или матрицах, содержащих несколько элементов или даже несколько десятков элементов. Хотя и в этом случае оценка их размеров полезна при разработке алгоритмов матричных вычислений и оценке их эффективности в части использования памяти.

Как отмечалось, в левой части окна системы MATLAB имеется окно специального браузера рабочей области — Workspace Browser. Он служит для просмотра ресурсов рабочей области памяти. Браузер дает наглядную визуализацию содержимого рабочей области. Окно браузера рабочей области выполняет и другие важные функции — позволяет просматривать существующие в памяти объекты, редактировать их содержимое и удалять объекты из памяти. Для вывода содержимого объекта достаточно выделить его имя с помощью мыши и щелкнуть на кнопке

Open (Открыть). Объект можно открыть и двойным щелчком на его имени в списке. Откроется окно редактирования массива Array Editor, показанное на рисунке 10, применительно к матрице M.

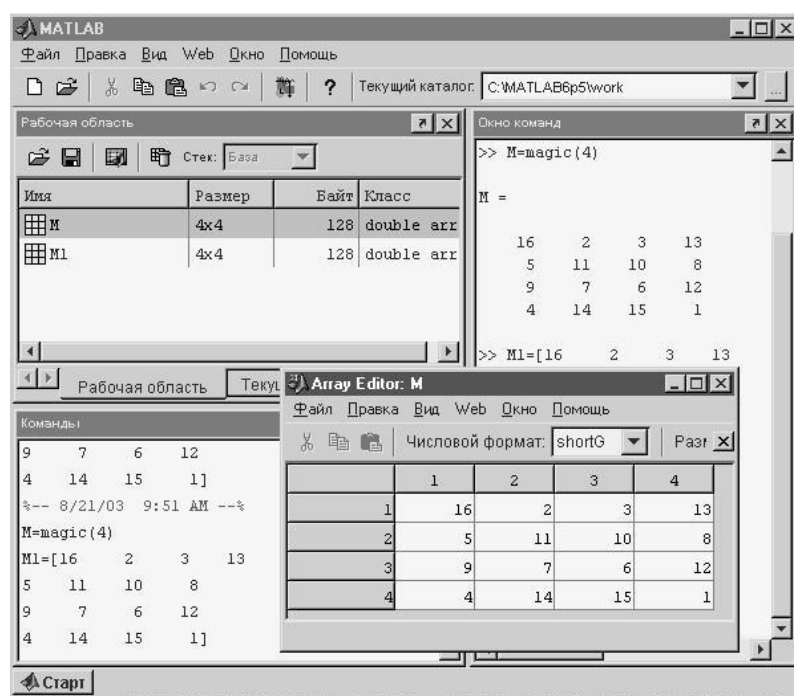


Рисунок 10 – Пример просмотра содержимого матрицы

Окно редактирования матрицы дает удобный доступ для редактирования любого элемента матрицы по правилам, принятым при работе с электронными таблицами. Основное из них — быстрый доступ к любому элементу матрицы. Можно также менять тип значений элементов, выбирая его из списка, предоставляемого меню Numeric format (Формат чисел). В окне также выводятся данные о числе строк и столбцов матрицы.

Браузер файловой структуры

Для просмотра файловой структуры MATLAB служит специальный браузер файловой системы (Path Browser), который запускается при обычной загрузке системы. Если был установлен упрощенный интерфейс, то для запуска браузера файловой системы используется окно Current Directory

(Текущая папка). На рисунке 11 в левой части показано окно этого браузера.

Нетрудно заметить, что браузер файловой системы построен по типу общеизвестного Проводника (Explorer) операционных систем Windows. Окно дает возможность просмотра файловой системы ПК и выбора любого файла. Для примера на рисунке 11 показан выбор m-файла функции натурального логарифма.

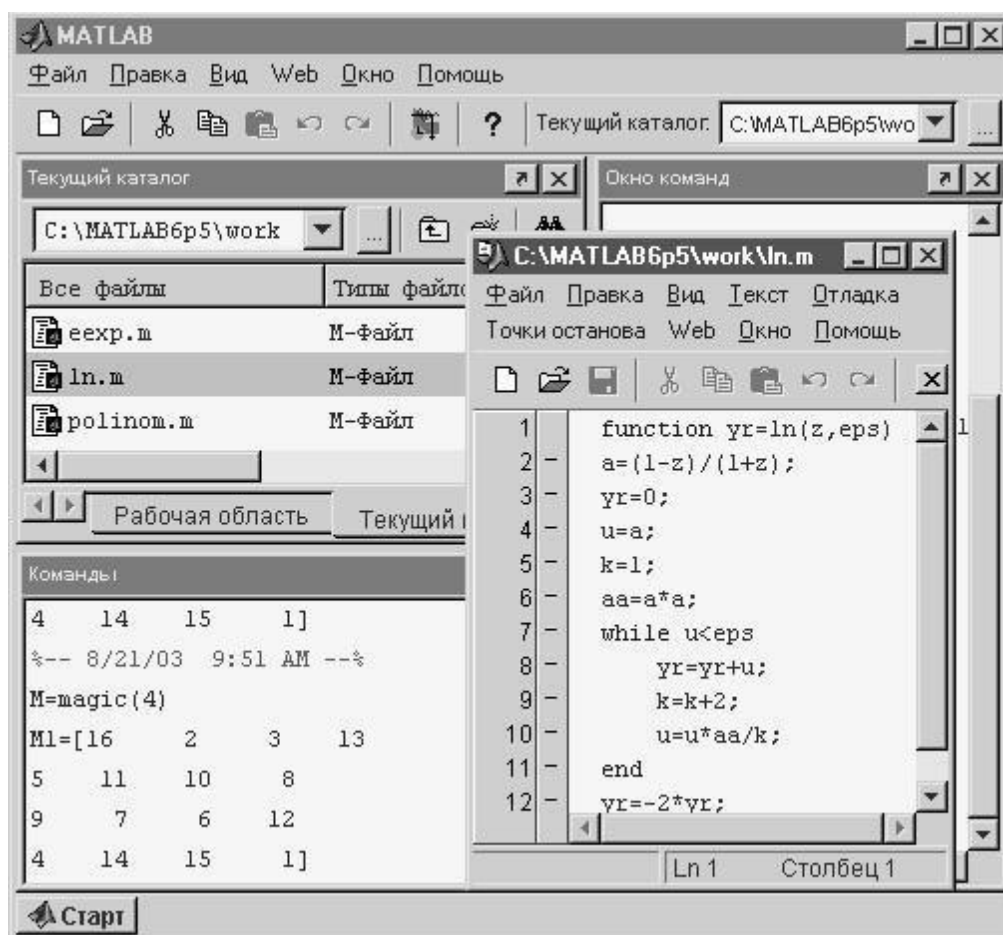


Рисунок 11 – Пример окна браузера

Исполнив команду Open (Открыть) из контекстного меню правой клавиши мыши или дважды щелкнув по выделенной строке с именем файла, можно ввести этот файл в окно редактора/отладчика m-файлов. При этом редактор запустится автоматически и его окно с готовым для редактирования выбранным файлом появится на экране – оно показано в центре экрана на рисунке 9.

Таким образом, браузер просмотра файловой структуры позволяет детально ознакомиться с файловой системой MATLAB и вывести любой из m-файлов или файлов демонстрационных примеров для просмотра, редактирования и модификации. **Работа с меню**

Команды, операции и параметры

Открытая позиция строки меню содержит различные операции и команды. Выделенная команда или операция выполняется при нажатии клавиши Enter (Ввод).

Выполнение команды можно также осуществить щелчком мыши или нажатием на клавиатуре клавиши, соответствующей выделенному символу в названии команды,

Между командами и операциями нет особых отличий, и в литературе по информатике их часто путают. Мы будем считать командой действие, которое выполняется немедленно. А операцией — действие, которое требует определенной подготовки, например открытие окна для установки определенных параметров.

Параметр (option)— это значение определенной величины, действующее во время текущей сессии. Параметрами обычно являются указания на применяемые наборы шрифтов, размеры окна, цвет фона и т. д.

Меню системы

Перейдем к описанию основного меню системы MATLAB. Это меню выглядит довольно скромно и содержит всего шесть пунктов:

- File — работа с файлами;
- Edit — редактирование сессии;
- View — вывод и скрывание панели инструментов;
- Web — доступ к Интернет-ресурсам; □ Windows — установка Windows-свойств окна;
- Help — доступ к справочным подсистемам.

Подменю File

Подменю File содержит ряд операций и команд для работы с файлами. Оно показано на рисунке 12. Число операций и команд значительно сокращено по сравнению с тем же меню у предшествующей версии системы MATLAB.

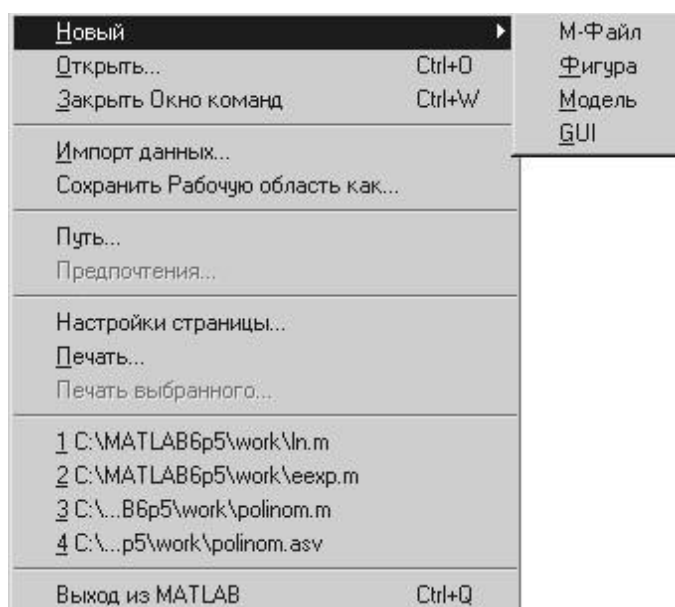


Рисунок 12 – Меню Файл

Теперь меню File содержит следующие операции:

- New - открывает подменю с позициями:
- M-file — открытие окна редактора/отладчика m-файлов;
- Figure — открытие пустого окна графики;
- Model — открытие пустого окна для создания Simulink-модели;
- GUI — открытие окна разработки элементов графического интерфейса пользователя.
- Open — открывает окно загрузки файла.
- Close Command Windows — закрывает окно командного режима работы

(оно при этом исчезает с экрана).

- Import data — открывает окно импорта файлов данных.
- Save Workspace As... — открывает окно записи рабочей области в виде файла с заданным именем.
- Set Path — открывает окно установки путей доступа файловой системы.
- Preferences... — открывает окно настройки элементов интерфейса.
- Print... — открывает окно печати всего текущего документа.
- Print Selection... — открывает окно печати выделенной части документа.
- Exit — завершает работу с системой.

Открытие окон для подготовки новых файлов

Команда New, как отмечено, открывает окна для подготовки новых файлов. Для трех типов файлов (m-файлы, графические файлы и файлы Simulink-моделей) окна их редакторов уже описывались. Новой является позиция GUI подменю File. Она открывает окно редактора элементов пользовательского интерфейса, показанное на рисунке 13.

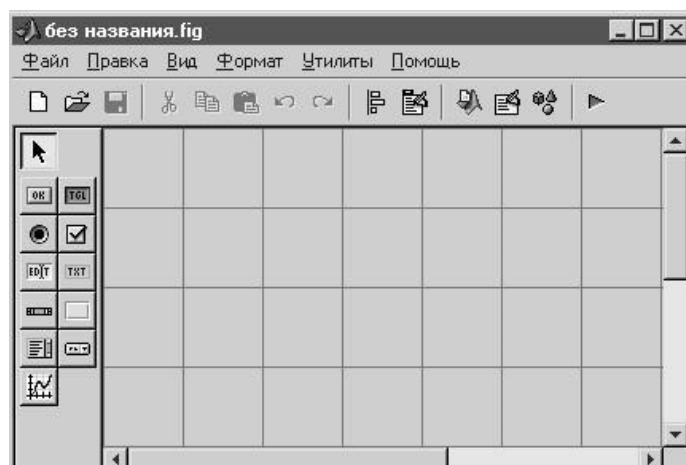


Рисунок 13 – Окно редактора элементов GUI

К подробному описанию этого окна мы еще вернемся. А пока отметим, что работа с редактором довольно очевидна.

Загрузка и сохранение файлов

Команды Open... (Открыть) и Import data... (Импортировать данные) выводят стандартные окна для загрузки m-файлов и файлов данных. Команда Save Workspace As... (Сохраните рабочую область как...) открывает стандартное окно записи файлов с расширением .mat. Они хранят определения переменных, массивов, функций пользователя и иных объектов, созданных в ходе текущей сессии работы. Эти команды в силу общеизвестности в более подробных комментариях не нуждаются.

Установка путей доступа файловой системы

Команда Set Path... (Установить путь) открывает окно редактора путей доступа файловой системы. Окно дает список папок с файлами системы MATLAB. Имеется возможность! переноса папок вверх или вниз по списку, уничтожения их и переименования. По умолчанию задается правильная установка путей доступа, так что данными возможностями стоит пользоваться только в особых обстоятельствах, например при! случайном переносе папок в другое место или при их переименовании.

Настройка элементов интерфейса

Окно настройки элементов интерфейса представлено на рисунке 14.

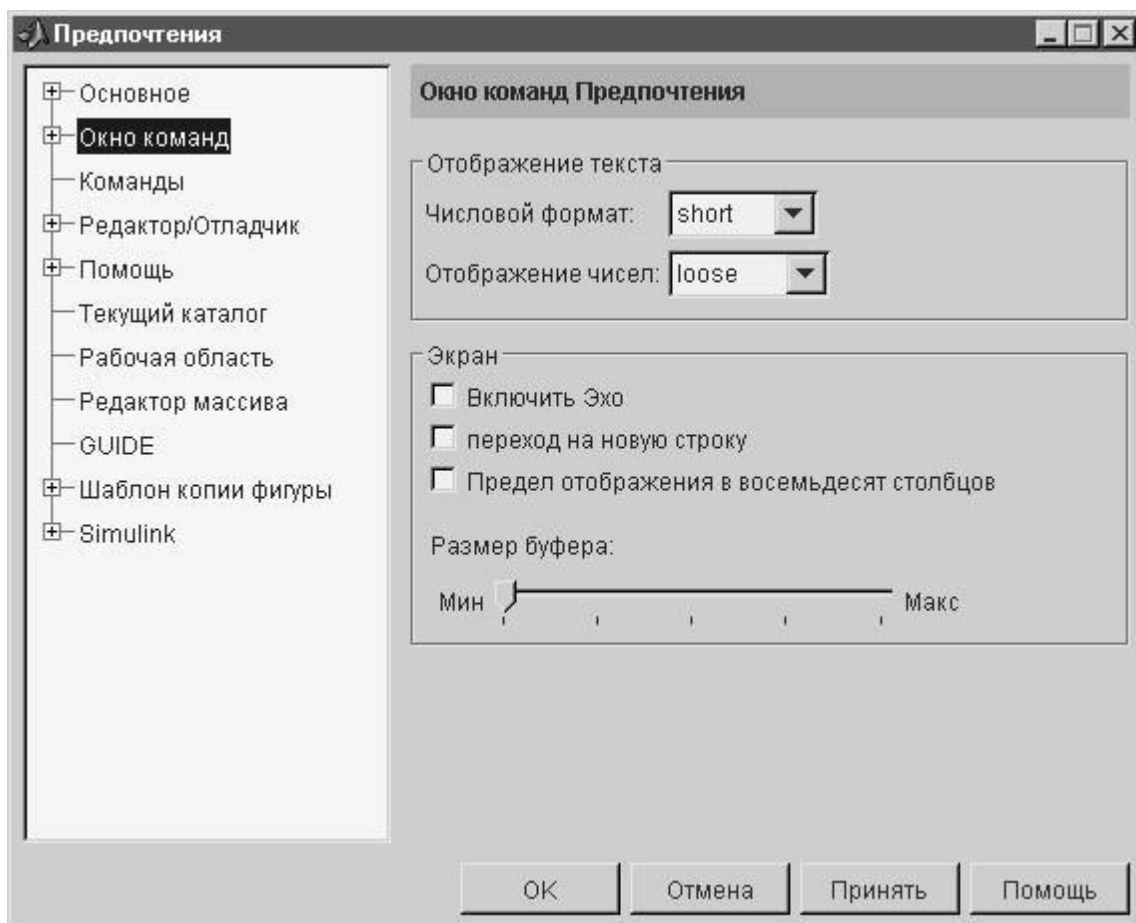


Рисунок 14 – Окно настройки элементов интерфейса

Интерфейс MATLAB стал настолько удобным и даже изысканным, что мало вероятно, что кому-либо захочется менять его в мере, большей, чем это предусмотрено командами подменю View. Однако такая возможность есть — команда Preferences... (Предпочтения) выводит окно детальной настройки элементов интерфейса. В левой части этого окна имеется древообразный список элементов интерфейса системы, а в правой части—поле задания параметров для выбранного типа элементов. Поскольку изменение параметров производится обычно очень редко, мы не будем детально рассматривать это окно. Заинтересованный читатель наверняка разберется с нужными ему параметрами самостоятельно.

Меню Edit — средства редактирования документов

Меню Edit (рисунок 15) содержит операции и команды редактирования, типичные для большинства приложений Windows: ☐ Undo (Отменить) — отмена результата предшествующей операции;

- ☐ Redo (Повторить) — отмена действия последней операции Undo;
- ☐ Cut (Вырезать) — вырезание выделенного фрагмента и перенос его в буфер;

- Copy (Копировать) — копирование выделенного фрагмента в буфер;
- Paste (Вставить) — вставка фрагмента из буфера в текущую позицию курсора;
- Clear (Очистить) — операция очистки выделенной области;
- Select All (Выделить) — выделение всей сессии;
- Delete (Стереть) — уничтожение выделенного объекта;
- Clear Command Windows (Очистить командное окно) — очистка текста сессии (с сохранением созданных объектов);
- Clear Command History (Очистить окно истории команд) — очистка окна истории;
- Clear Workspace — очистка окна браузера рабочей области.

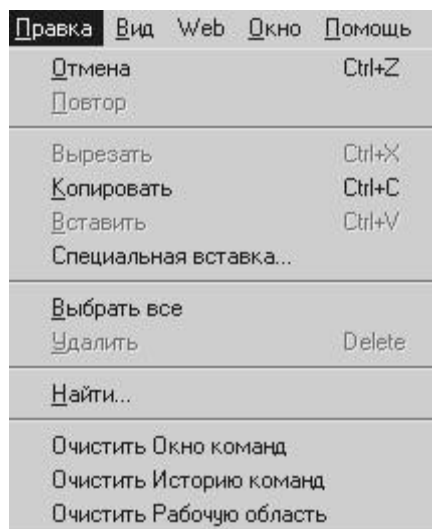


Рисунок 15 – Меню Edit

Назначение указанных команд и операций уже обсуждалось. Отметим лишь, что команда Clear Command Window очищает окно командного режима работы и помещает курсор в верхний левый угол окна. Однако все определения, сделанные в течение стертых таким образом сессий, сохраняются в памяти компьютера. Для очистки экрана используется также команда clc, вводимая в командном режиме.

Меню View и Window

В MATLAB что уже описывалось, набор команд меню View существенно расширен, и теперь с помощью этого меню можно существенно менять вид пользовательского интерфейса.

Меню Window активно только в случае, если в систему загружены файлы. При этом оно имеет единственную команду Close All (закрыть все окна) и открывающийся

список всех загруженных файлов. Он позволяет выбрать окно указанного пользователем файла и сделать его открытым.

Основы редактирования и отладки m-файлов **Интерфейс редактора/отладчика m-файлов**

Для подготовки, редактирования и отладки m-файлов служит специальный многооконный редактор. Он выполнен как типичное приложение Windows. Редактор можно вызвать командой `edit` из командной строки или командой `New→M-file` меню `File`. После этого в окне редактора можно создавать свой файл, пользоваться средствами его отладки и запуска. Перед запуском файла его необходимо записать на диск, используя команду `File→Save as` в меню редактора.

На рисунке 16 показано окно редактора/отладчика с текстом простого файла в окне редактирования и отладки.

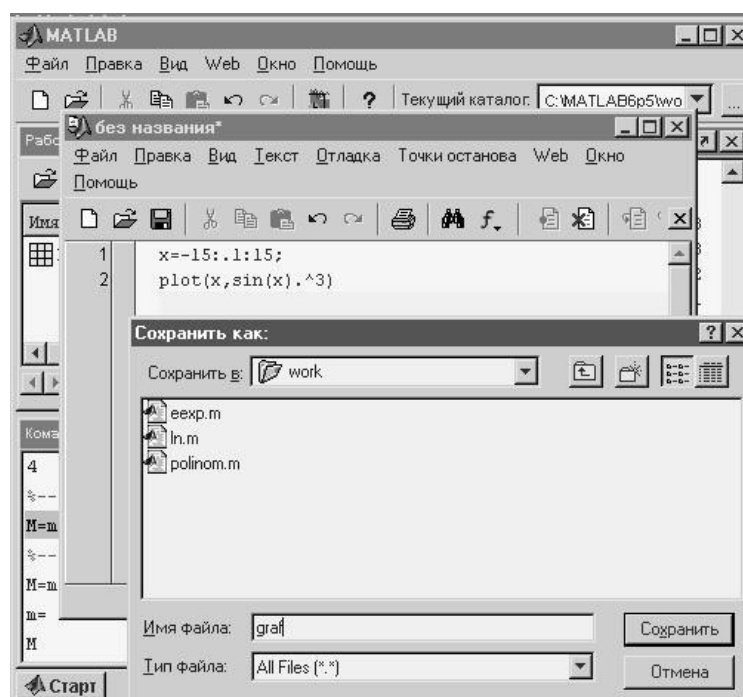


Рисунок 16 – Редактор/отладчик файлов при записи файла на диск

Подготовленный текст файла (это простейшая и наша первая программа на языке программирования MATLAB) надо записать на диск. Для этого используется команда `Save As`, окно которой видно на рис. 16 внутри окна системы редактора/отладчика. Работа с окном команды `Save As` уже описывалась.

После записи файла на диск можно заметить, что команда `Run` в меню `Tools` (Инструменты) редактора становится активной (до записи файла на диск она пассивна) и позволяет произвести запуск файла. Запустив команду `Run`, можно наблюдать исполнение m-файла.

Для удобства работы с редактором/отладчиком строки программы в нем нумеруются в последовательном порядке. Редактор является многооконным. Окно каждой программы оформляется как вкладка. Цветовые выделения и синтаксический контроль.

Редактор/отладчик m-файлов выполняет синтаксический контроль программного кода по мере ввода текста. При этом используются следующие цветовые выделения:

- ключевые слова языка программирования — синий цвет;
- операторы, константы и переменные — черный цвет;
- комментарии после знака % — зеленый цвет;
- символьные переменные (в апострофах) — зеленый цвет; □ синтаксические ошибки — красный цвет.

Благодаря цветовым выделениям вероятность синтаксических ошибок снижается,

Панель инструментов редактора и отладчика

Редактор имеет свое меню и свою инструментальную панель. Внешний вид инструментальной панели показан на рисунке 17. По стилю данная панель похожа на панель инструментов окна командного режима работы, но имеет несколько иной набор кнопок.



Рисунок 17 – Панель инструментов редактора/отладчика m-файлов

Назначение кнопок панели инструментов редактора/отладчика следующее:

- New — создание нового m-файла;
- Open — вывод окна загрузки файла;
- Save — запись файла на диск;
- Print — печать содержимого текущего окна редактора;
- Cut — вырезание выделенного фрагмента и перенос его в буфер;
- Copy — копирование выделенного объекта в буфер;
- Paste — размещение фрагмента из буфера в позиции текстового курсора;
- Undo — отмена предшествующей операции;
- Redo — повтор отмененной операции;

- Find text — нахождение указанного текста;
- Show function — показ функции;
- Set/Clear Breakpoint — установка/сброс точки прерывания;
- Clear All Breakpoints — сброс всех точек прерывания;
- Step — выполнение шага трассировки;
- Step In — пошаговая трассировка с заходом в вызываемые т-файлы;
- Step Out — пошаговая трассировка без захода в вызываемые т-файлы;
- Save and Run — запись и сохранение;
- Exit Debug Mode — выход из режима отладки.

С назначением ряда из этих кнопок вы уже знакомы, поскольку оно аналогично описанному ранее для основного окна MATLAB. А вот о назначении других кнопок надо поговорить.

Работа с точками прерывания

Основным приемом отладки m-файлов является установка в их тексте точек прерывания (Breakpoints). Они устанавливаются (и сбрасываются) с помощью кнопок Set/Clear Breakpoint. Сброс всех точек прерывания обеспечивается кнопкой Clear All Breakpoints.

Вы можете задать выполнение программы без остановки при заходе, но с остановкой при выходе (кнопка Step Out), и с остановкой при заходе в вызываемые m-файлы (кнопка Step In). Кнопка Exit Debug Mode (Выход из режима отладки) прекращает операции отладки.

3. Простейшие математические вычисления

MATLAB — это математический пакет прикладных программ, основанный на использовании матриц. Пакет содержит обширную библиотеку программ по численным методам, использует двух- и трехмерную графику, а также форматы языков высокого уровня. Подходящий формат позволяет с помощью MATLAB быстро выполнять и модифицировать программы.

Арифметические операции

+	Сложение
—	Вычитание
*	Умножение
/	Деление
^	Возведение в степень

π , e , i Постоянные

Пример

```
>>(2+3*pi)/2
```

```
ans = 5.7124
```

Встроенные функции

Ниже кратко перечислены некоторые функции, имеющиеся в MATLAB. Следующий пример иллюстрирует, как можно комбинировать функции и арифметические операции. Описание других функций можно найти, используя легко доступную справку в диалоговом режиме: `abs(#) cos(#) exp(#) log(#) log10(#) cosh(#) sin(#) tan(#) sqrt(#) floor(#) acos(#) tanh(#)`.

Пример

```
>>3*cos(sqrt(4.7)) ans = -1.6869
```

По умолчанию выводится приблизительно пять десятичных значащих цифр. Команда `format long` позволяет вывести приблизительно 15 десятичных значащих цифр.

Пример

```
>>format long >>3*cos(sqrt(4.7)) ans = -1.68686892236893
```

Операторы присваивания

Для присваивания выражениям различных имен используется знак равенства.

Пример

```
>>a=3-floor(exp(2.9)) a = -15
```

Точка с запятой, расположенная в конце выражения, предполагает вывод символа на экран компьютера (выход).

Пример

```
>>b=sin(a); >>2*b^2 ans=0.8457
```

Определение функций

В MATLAB пользователь может определить функцию, построив М-файл (в конце файла ставится `.m`) в М-файле Editor/Debugger. Однажды определенная пользователем функция вызывается таким же способом, как и встроенные функции.

Пример

Определим функцию $\text{fun}(x)=1+x-x^2/4$ в m-файле `fun.m`. В Editor/Debugger она записывается следующим образом.

```
function y=fun(x) y=1+x-x.^2/4;
```

Для обозначения переменных можно употреблять различные буквы и для названия функций — различные имена, но следует использовать один и тот же формат. Функцию, однажды записанную как М-файл под именем fun.m, можно вызывать в MATLAB Command Window так же, как любую другую функцию.

Пример

```
>>cos(fun(3)) ans= -0.1782
```

Полезным и эффективным способом вычисления функций является использование команды feval. Она требует, чтобы функция вызывалась, как строка.

Пример

```
>>feval('fun',4) ans=1
```

Матрицы

Все переменные в MATLAB интерпретируются как матрицы или массивы. Матрицы можно вводить непосредственно:

Пример >>A=[1 2 3; 4 5 6; 7 8 9]

A= 1 2 3

4 5 6

7 8 9

Точка с запятой используется для разделения строк матрицы. Отметим,

что элементы матрицы следует разделять одиночным пробелом.

Альтернативный способ ввода матрицы — строка за строкой.

Пример >>A=[1 2 3

4 5 6

7 8 9]

A= 1 2 3

4 5 6

7 8 9

Матрицы можно формировать, используя встроенные функции.

Пример >>Z=zeros(3,5); Создание нулевой матрицы размера 3x5.

>>X=ones(3,5); Создание матрицы размера 3x5, состоящей из единиц.

>>Y=0:0.5:2 Вывод на экран матрицы размера 1x5.

Y=0 0.5000 1.0000 1.5000 2.0000

Y >>cos (Y) Создание матрицы размера 1x5, каждый элемент которой является cos от

ans=1.0000 0.8776 0.5403 0.0707 -0.4161

Элементами матрицы можно манипулировать несколькими способами. Пример

>>A (2,3) Выделение одного элемента матрицы A ans = 6

>>A (1:2, 2:3) Выделение подматрицы A ans = 2 3 5 6

>>A([1 3],[1 3]) Другой способ выделения подматрицы A ans= 1 3
7 9

>>A(2,2) =tan(7.8);

Присвоение нового значения элементу матрицы A

Дополнительные команды для матриц можно найти, используя справку в диалоговом режиме или документацию к пакету прикладных программ.

Операции с матрицами

+ Сложение

– Вычитание

* Умножение

^ Возведение в степень

' Сопряжение и транспонирование

Пример >>B=[1 2; 3 4];

>>C=B' C равна транспонированной матрице B

C= 1 3

2 4

>>3*(B*C)^3 ans= 13080 29568 29568 66840

Операции с массивами

Одним из наиболее полезных свойств пакета MATLAB является количество операций, которые можно выполнять над отдельными элементами матрицы. Такая операция была продемонстрирована выше, когда элементы матрицы размера 1x5 записывались как cos от элементов вводимой матрицы. Операции сложения, вычитания и умножения матрицы на скаляр всегда осуществляются поэлементно, но это не относится к операциям умножения, деления и возведения матрицы в степень. Данные три операции можно производить поэлементно, как и предыдущие, но с указателем операции: .*, ./ и .^. Важно понять, как и когда именно их можно использовать. Операции над массивами являются решающими для эффективного построения и выполнения пакетом MATLAB программ и графиков.

Пример >>A=[1 2;3 4];

```
>>A^2 Произведение матриц AA ans= 7 10 15 22
```

```
>>A.^2 Квадрат каждого элемента A ans= 1 4 9 16
```

```
>>cos(A./2)
```

```
Деление каждого элемента A на 2 и вычисление cos от каждого элемента ans=  
0.8776 0.5403 0.0707 -0.4161
```

Графики

Пакет MATLAB позволяет строить двух- и трехмерные графики кривых и поверхностей. Дополнительные возможности и описание графиков в пакете MAT-LAB можно найти, используя справку в диалоговом режиме или документацию к пакету прикладных программ.

Команду plot используют для построения графика двумерных функций. Следующий пример показывает, как создать программу построения графика функций $y=\cos(x)$ и $y=\cos^2(x)$ на интервале $[0; \pi]$.

Пример

```
>>x=0:0.1:pi;
```

```
>>y=cos(x);
```

```
>>z=cos(x).^2;
```

```
>>plot(x, y, x, z, 'o')
```

В первой строке программы задается область с шагом 0,1. В следующих двух строках задаются две функции. Заметим, что первые три строки заканчиваются точкой с запятой. Точка с запятой необходима, чтобы запретить вывод матриц x , y и z на экран. В четвертой строке содержится команда plot, которая и строит график. Первые параметры команды plot, x и y , строят график функции $y=\cos(x)$. Третий и четвертый параметры, x и z , строят график функции $y=\cos^2(x)$. Последний параметр – 'o'; в нем содержится результат построения графика в каждой точке (x_k, z_k) , где $z=\cos^2(x_k)$.

В третьей строке необходимо использовать оператор ".", так как сначала вычисляется значение cos каждого элемента матрицы x . Затем каждый элемент матрицы $\cos(x)$ с помощью команды .^ возводится в квадрат.

Команда построения графика fplot — это полезная альтернатива команде plot. Форма ее записи — fplot ('name', [a, b], n). Она строит график функции name.m по множеству n точек на интервале $[a; b]$. По умолчанию число n равно 25.

Пример

```
>>fplot('tanh',[-2,2])
```

График $y=\tanh(x)$ на $[-2; 2]$

Команды plot и plot3 соответственно используются для построения параметрических кривых в двух- и трехмерном пространстве. Они особенно полезны для

создания зрительных образов решений дифференциальных уравнений в двух и трех измерениях.

Пример Построим эллипс $c(t) = (2\cos(t), 3\sin(t))$, где $0 < t < 2\pi$, используя следующие команды

```
>>t=0:0.2:2*pi;  
>>plot (2*cos(t),3*sin(t))
```

Пример График кривой $c(t)=(2\cos(t), t^2, 1/t)$, $0.1 < t < 4\pi$, получаем с помощью следующих команд

```
>>t=0.1:0.1:4*pi;  
>>plot3(2*cos(t),t.^2,1./t);
```

Графики трехмерных поверхностей получаем, задав подмножество прямоугольной области определения функции с помощью команды `meshgrid`, и затем используем команду `mesh` или `surf` для построения графика. Эти графики полезны для создания зрительных образов решений дифференциальных уравнений в частных производных.

Пример

```
>>x=-pi:0.1:pi;  
>y=x;  
>>[x ,y]=meshgrid(x,y);  
>>z=sin(cos(x+y));  
>>mesh(z)
```

Циклы и условные операторы Операторы отношений

`==` Равно

`~=` Не равно

`<` Меньше

`>` Больше

`<=` Меньше или равно

`>=` Больше или равно

Логические операторы

`~` Not (дополнение)

`&` And (справедливо, если справедливы оба операнда)

`|` Or (справедливо, если один из двух или оба операнда справедливы)

Булевы величины

1 True

0 False

Операторы for, if и while установлены в MATLAB аналогично их прототипам в других языках программирования. Они имеют следующую основную форму.

for (переменная цикла = выражение цикла) Выполняемый оператор end
if (логическое выражение) Выполняемые операторы else (логическое выражение) Выполняемые операторы end
while (выражение условия) Выполняемые операторы end

В следующем примере показано, как использовать вложенные циклы для построения матрицы. Запишем файл как М-файл под именем nest.m.

Введя nest в MATLAB Command Window, можно создать матрицу A.

Заметим, что если смотреть из верхнего левого угла, то элементы матрицы A – это элементы треугольной матрицы Паскаля.

Пример

```
for i=1:5
    A(i, 1)=1;
    A(1, i)=1;
end
for i=2:5
    for j=2:5
        A(i, j)=A(i, j-1)+A(i-1, j);
    end
end
```

A

Команда break используется для выхода из цикла.

Пример

```

for k=1:100
x=sqrt(k);
if ((k>10)&(x-floor(x)==0))
break
end
end
k

```

Команду disp можно использовать для вывода на дисплей текста или матрицы.

Пример

```

n=10;
k=0;
while k<=n x=k/3;
disp([x x^2 x^3]) k=k+1;
end

```

Программы

Эффективным способом создания программ является использование определенных программистом функций. Их записывают в виде М-файлов.

Эти программы разрешают пользователю задавать ввод и вывод параметров. Они легко вызываются, как подпрограммы в других программах. Приведенный ниже пример позволяет увидеть результат построения треугольника Паскаля по первому числу. Введем следующую функцию в MATLAB Editor/Debugger и запишем ее как М-файл под именем pasc.m.

```

Пример      function P = pasc(n, m)
% Вход  - n -- количество строк %      - m -- первое число
% Выход – P -- треугольник Паскаля for j=1:n P(j, 1)=1;
P(1, j)=1;
End for k=2: n for j=2: n
P(k,j)=rem(P(k,j-D,m)+rem(P(k-1,j) ,m);
end
end

```

Теперь в MATLAB Command Window введем P=pasc(5,3), чтобы увидеть первые пять строк треугольника Паскаля по модулю 3, или попытаемся построить P=pasc(175,3);

(обратите внимание на точку с запятой). Введем $\text{sru}(P)$ (строится разреженная матрица для больших значений n).

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

На m -языке Matlab написать функцию, которая:

1. Возвращает матрицу элементы которой $a_{ij}=i+j$.
2. Возвращает матрицу элементы которой $a_{ij}=i-j$.
3. Возвращает матрицу элементы которой $a_{ij}=i*j$.
4. Решает квадратное уравнение.
5. Вычисляет значение полинома, заданного коэффициентами.
6. Выводит на экран название Вашей специальности.
7. Возвращает матрицу элементы которой $a_{ij}=2^{i+j}$.
8. Вычисляет функцию логической неравнозначности.
9. Вычисляет логическую функцию стрелка Пирса.
10. Вычисляет логическую функцию штрих Шеффера.
11. Сравнивает два числа.
12. Вычисляет кубический корень.
13. Разворачивает матрицу в вектор.
14. Выводит на экран Вашу фамилию, имя и отчество.
15. Возвращает матрицу элементы которой пронумерованы по спирали, начиная с центра.
16. Подсчитывает количество нулевых элементов матрицы.
17. Возвращает сумму элементов матрицы.
18. Подсчитывает количество отрицательных элементов матрицы.
19. Транспонирует матрицу.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая версия программы Matlab установлена на Вашем компьютере?
2. В чем отличие версии Matlab установленной на компьютер, от описанной?

3. Какие пакеты программы Matlab установлена на Вашем компьютере?

ПРАКТИЧЕСКАЯ РАБОТА 2. Пакет моделирования нейронных сетей neural network toolbox

Цель занятия:

Изучить интерфейс пакета моделирования нейронных сетей и получить практические навыки в создании компьютерных моделей искусственных нейронных сетей.

Подготовка к занятию:

Изучить по рекомендованной литературе и конспекту лекций основные понятия искусственных нейронных сетей, иметь практические навыки в использовании пакета компьютерной математик для реализации простейших математических вычислений.

Порядок проведения занятия

1. Графический интерфейс пакета моделирования нейронных сетей.
2. Создание программных моделей нейронных сетей.

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ

1. Графический интерфейс пакета моделирования нейронных сетей

Отличительной особенностью 12-го выпуска программных продуктов фирмы Math-Works является включение в их состав различных инструментальных средств организации диалога с пользователем. Как правило, это GUI-интерфейсы не является исключением и пакет по нейронным сетям ППП NNT, в состав которого входит инструментальное средство NNTool Этот графический интерфейс позволяет, не обращаясь к командному окну системы MATLAB, выполнять создание, обучение, моделирование, а также импорт и экспорт нейронных сетей и данных, используя только инструментальные возможности GUI-интерфейса Конечно, такие инструменты наиболее эффективны лишь на начальной стадии работы с пакетом, поскольку имеют определенные ограничения. В частности, интерфейс NNTool допускает работу только с простейшими однослойными и двухслойными нейронными сетями, но при этом пользователь выигрывает во времени и эффективности освоения новых объектов.

Вызов GUI-интерфейса NNTool возможен либо командой `nntool` из командной строки, либо из окна запуска приложений Launch Pad с помощью опции NNTool из раздела Neural Network Toolbox. После вызова на экране терминала появляется окно Network/Data Manager (Управление сетью/данными) (рисунок 1).

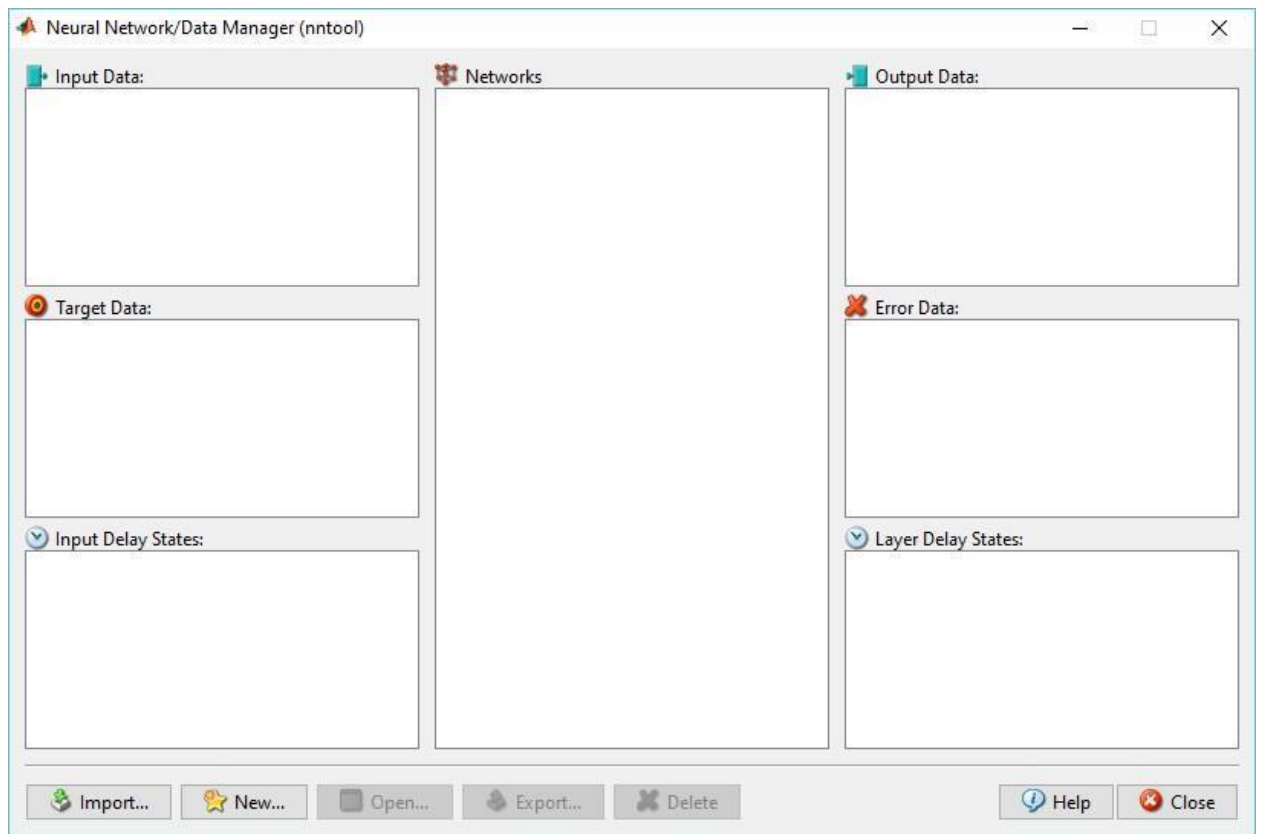


Рисунок 1 □ Общий вид окна Network/Data Manager

Здесь:

Help - кнопка вызова окна подсказки Network/Data Manager Help
(рисунок 2),

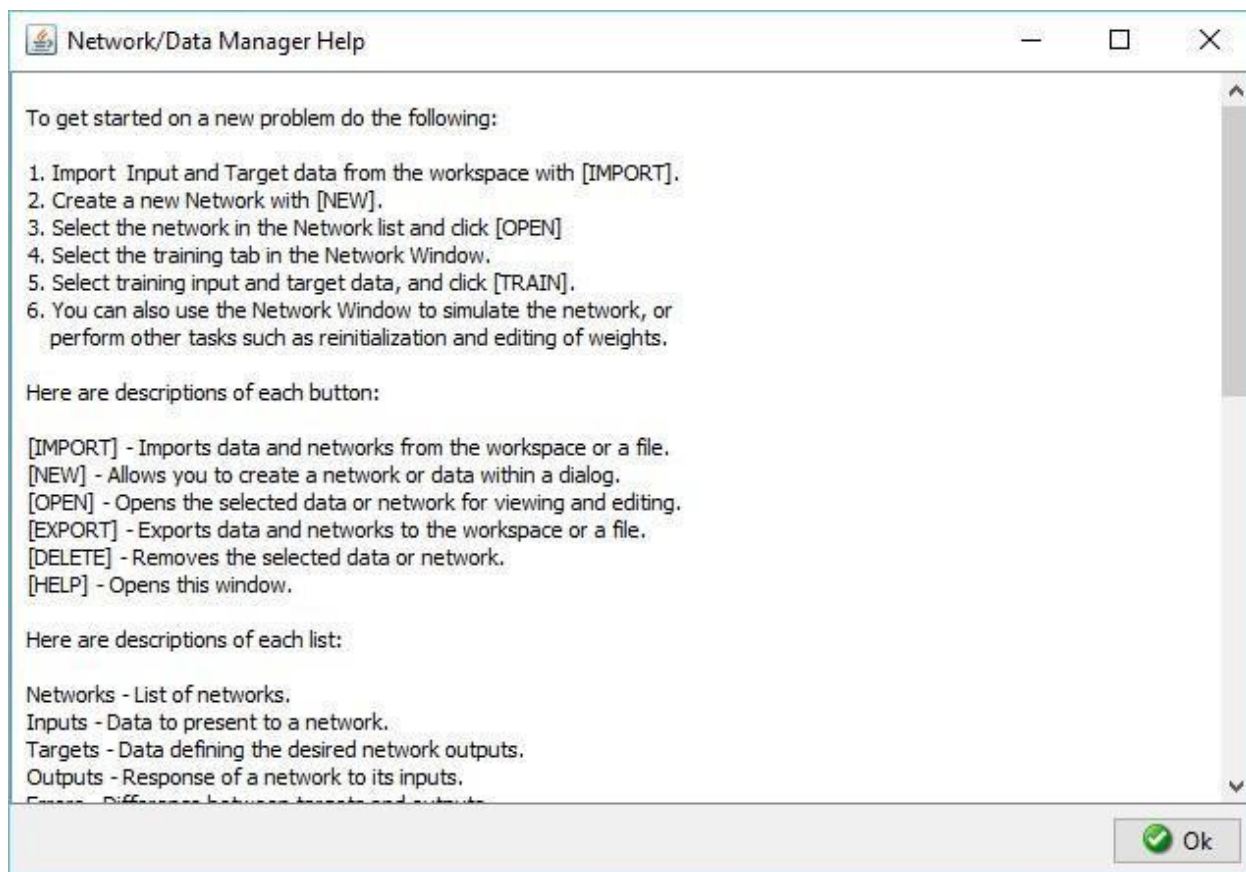


Рисунок 2 □ Окно подсказки Network/Data Manager Help

New Data... - кнопка вызова окна формирования данных Create New Data (рисунок 3),

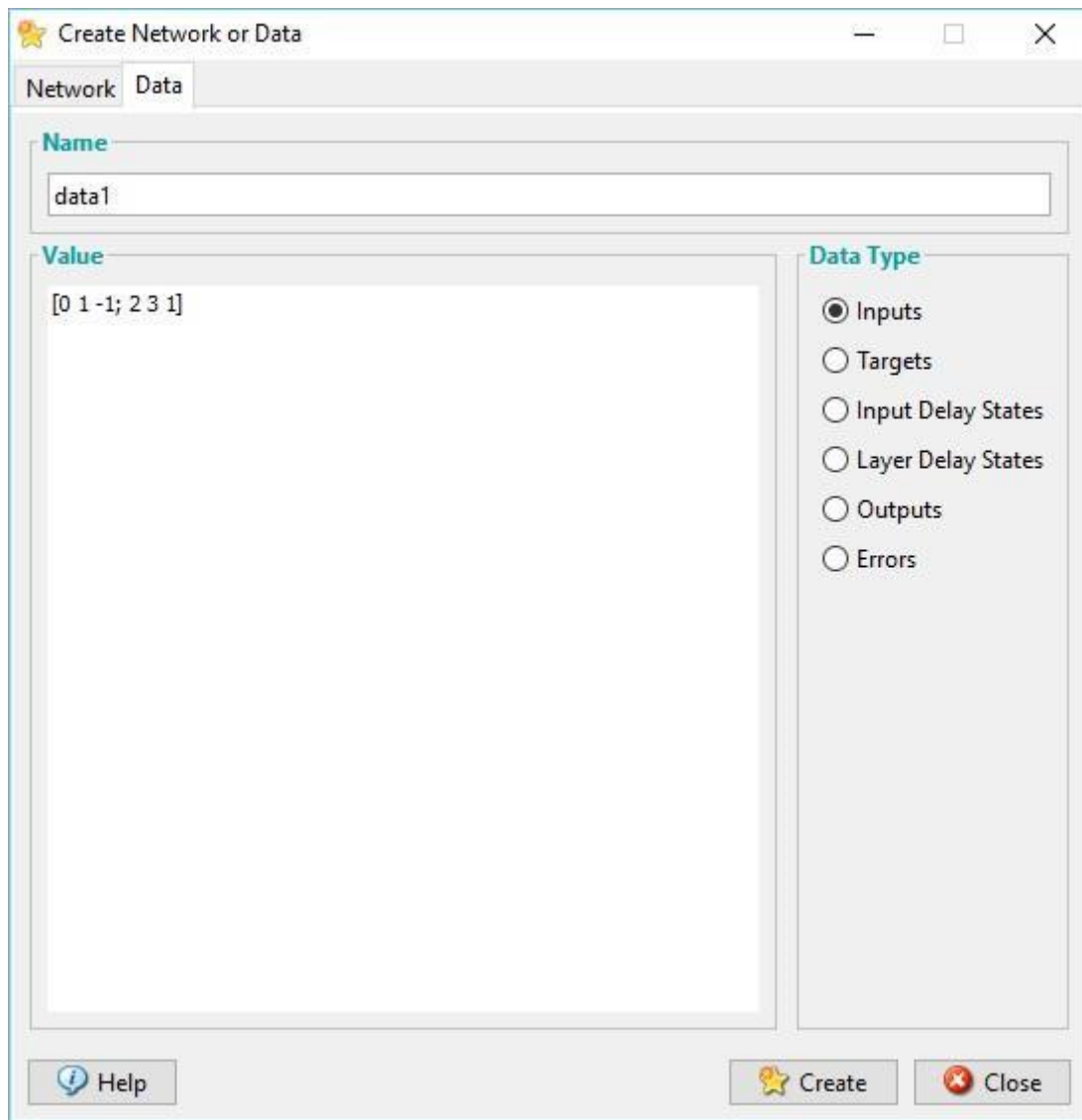


Рисунок 3 □ Окно формирования данных

New Network... - кнопка вызова окна создания новой нейронной сети

Create New Network (рисунок 4),

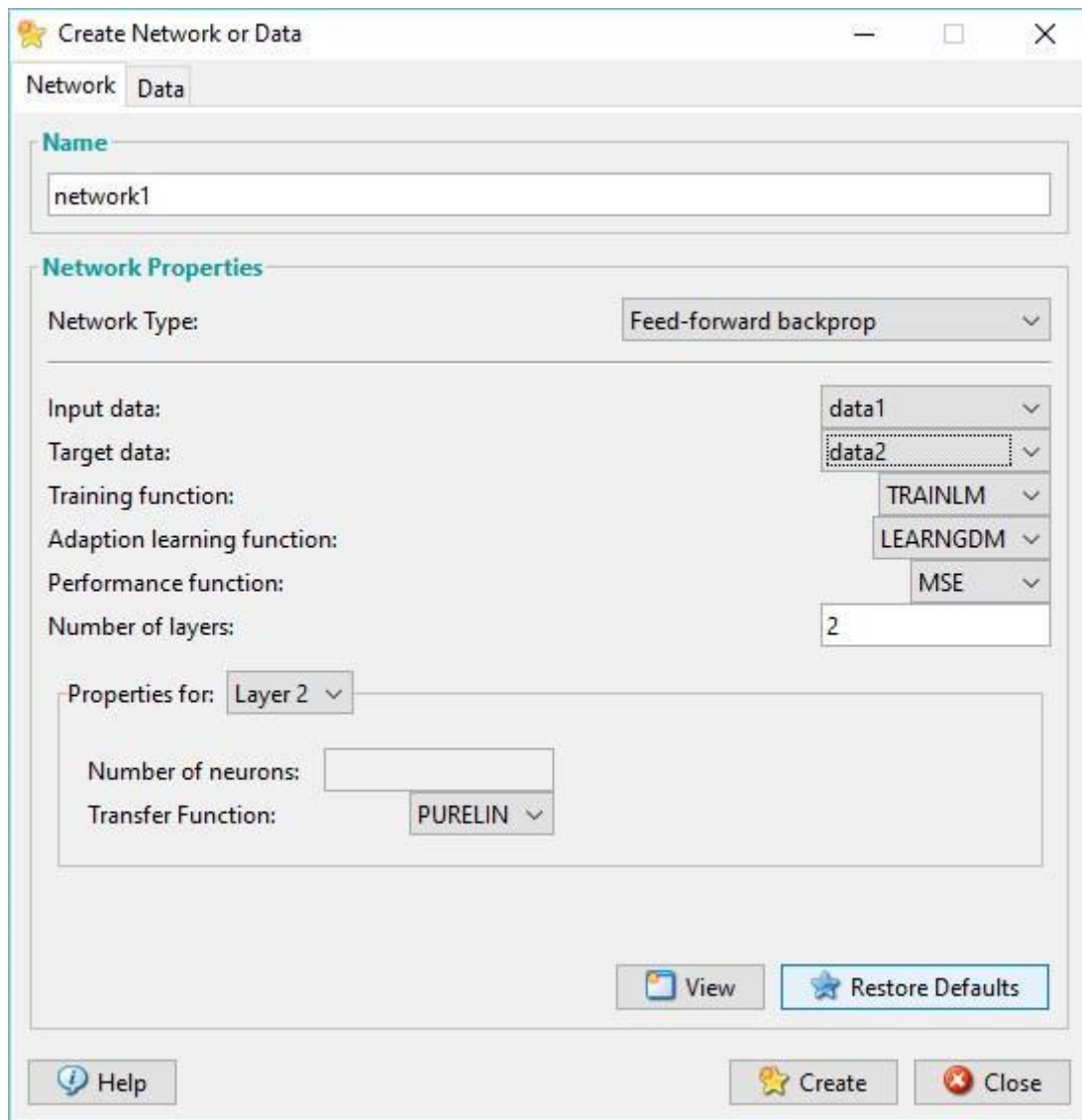


Рисунок 4 □ Окно создания новой нейронной сети

Import... - кнопка вызова окна для импорта или загрузки данных Import or Load to Network/Data Manager (рисунок 5),

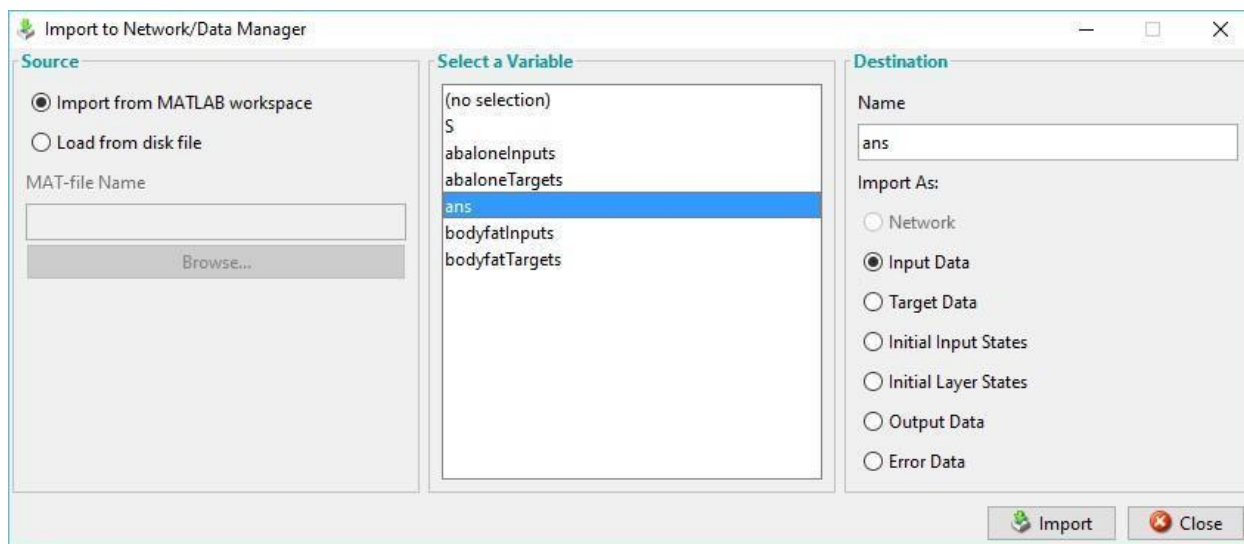


Рисунок 5 □ Окно для импорта или загрузки данных

Export... - кнопка вызова окна для экспорта или записи данных в файл Export or Save from Network/Data Manager (рисунок 6).

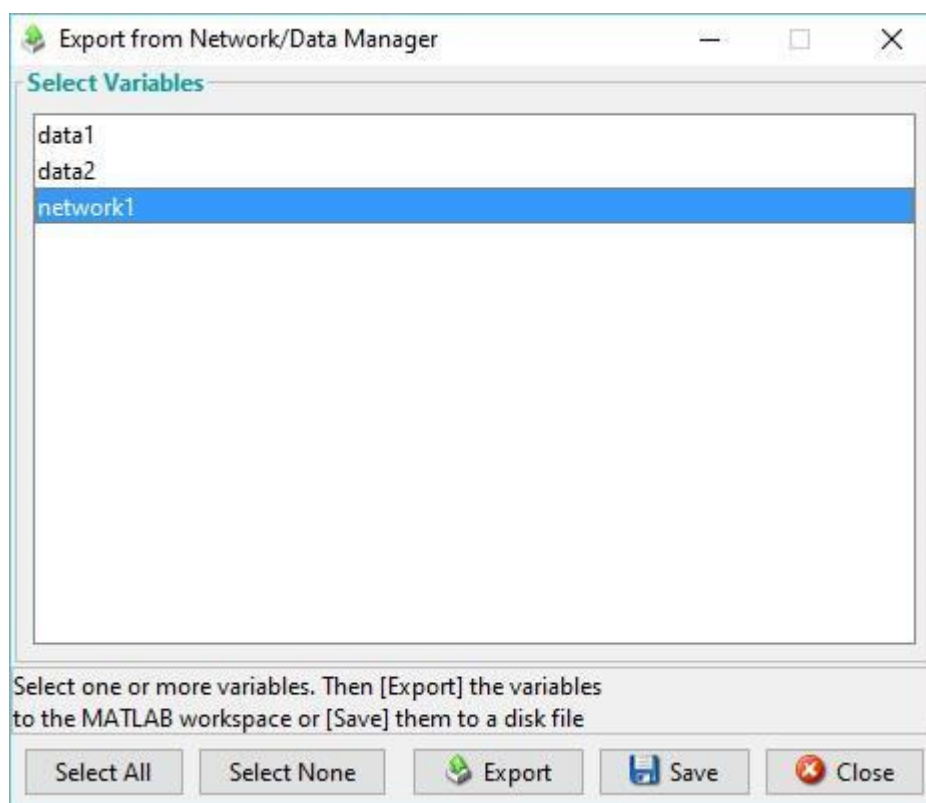


Рисунок 6 □ Окно для экспорта или записи данных в файл

Кнопки View, Delete становятся активными только после создания и активизации данных, относящихся к последовательностям входа, цели, выхода или ошибок сети. Кнопка View позволяет просмотреть, а кнопка

Delete удалить активизированные данные.

Кнопки View, Delete, Initialize..., Simulate..., Train..., Adapt... становятся активными после создания и активизации самой нейронной сети. Они позволяют просмотреть, удалить, инициализировать, промоделировать, обучить или адаптировать нейронную сеть и будут описаны ниже.

Прежде всего, рассмотрим назначение и способы работы с перечисленными выше окнами:

Окно Network/Data Manager Help. Это окно подсказки показано на рисунке 2 и описывает правила работы с диспетчером Network/Data Manager при создании нейронной сети.

Чтобы создать нейронную сеть, необходимо выполнить следующие операции:

- Сформировать последовательности входов и целей (кнопка New Data), либо загрузить их из рабочей области системы MATLAB или из файла (кнопка Import).
- Создать новую нейронную сеть (кнопка New Network), либо загрузить ее из рабочей области системы MATLAB или из файла (кнопка Import).
- Выбрать тип нейронной сети и нажать кнопку Train..., чтобы открыть окно для задания параметров процедуры обучения
- Открыть окно Network для просмотра, инициализации, моделирования, обучения и адаптации сети.

Окно Create New Data. Это окно показано на рисунке 3 и включает 2 области редактирования текста для записи имени вводимых данных (область Name) и ввода самих данных (область Value), а также 6 кнопок для указания типа вводимых данных.

Различают следующие типы данных:

- Inputs (Входы) - последовательность значений входов;
- Targets (Цели) - последовательность значений цели;
- Input Delay States (Состояния ЛЗ входа) - начальные условия линии задержки на входе;
- Layer Delay States (Состояния ЛЗ слоя) - начальные условия линии задержки в слое;
- Outputs (Выходы) - последовательность значений выхода сети;
- Errors (Ошибки) - разность значений целей и выходов.

Как правило, пользователь задает только последовательности входа и цели, т. е. типы данных Inputs и Targets. При этом следует помнить, что при адаптации нейронной сети данные должны быть представлены в виде массива ячеек.

Окно Create New Network. Это окно показано на рисунок 4 и включает поля для задания параметров создаваемой сети. В зависимости от типа сети количество полей и их названия изменяются.

Обратимся к описанию полей.

Network Name (Имя сети) - стандартное имя сети, присваиваемое GUI интерфейсом NNTool, в процессе создания новых сетей порядковый номер будет изменяться автоматически.

Network Type (Тип сети) - список сетей, доступных для работы с интерфейсом NNTool Для удобства этот список повторен в нижеследующей таблице Интерфейс NNTool позволяет создавать нейронные сети только с одним или двумя слоями.

Input ranges (Диапазоны входа) - допустимые границы входов, которые либо назначаются пользователем, либо определяются автоматически по имени входной последовательности, выбираемой из списка Get from Input...

Training function (Функция обучения) - список обучающих функций Adaption learning function (Функции настройки для режима адаптации) – список функций настроек.

Performance function (Функция качества обучения) - список функций оценки качества обучения.

Number of layers (Количество слоев) - количество слоев нейронной сети Properties for (Свойства) - список слоев Layer 1 (Слой 1), Layer 2 (Слой 2) Number of neurons (Количество нейронов) - количество нейронов в слое Transfer function (Функция активации) - функция активации слоя.

Окно Import or Load to Network/Data Manager. Это окно показано на рисунке 15 и включает 3 поля:

Source (Источник) - поле для выбора источника данных Это либо рабочая область системы MATLAB (кнопка выбора Import from MATLAB Workspace), либо файл (кнопка выбора Load from disk file).

Если выбрана первая кнопка, то в поле Select a Variable вы можете видеть все переменные рабочей области и, выбрав одну из них, можете определить ее в поле Destination (Назначение) как последовательность входа Inputs (Входы).

Если выбирается кнопка Load from disk file, то активизируется поле MAT-file Name и кнопка Browse, что позволяет начать поиск и загрузку файла из файловой системы.

Окно Export or Save from Network/Data Manager Это окно показано на рис 16 и позволяет передать данные из рабочей области GUI интерфейса NNTool в рабочую область системы MATLAB или записать их в виде файла на диске.

В данном случае нами выбрана переменная network1, которая принадлежит к классу network object и описывает нейронную сеть. После того как эта переменная

экспортирована в рабочую область, можно, например, построить модель нейронной сети в системе Simulink с помощью оператора gensim.

Диалоговая панель Network показана на рисунок 7.

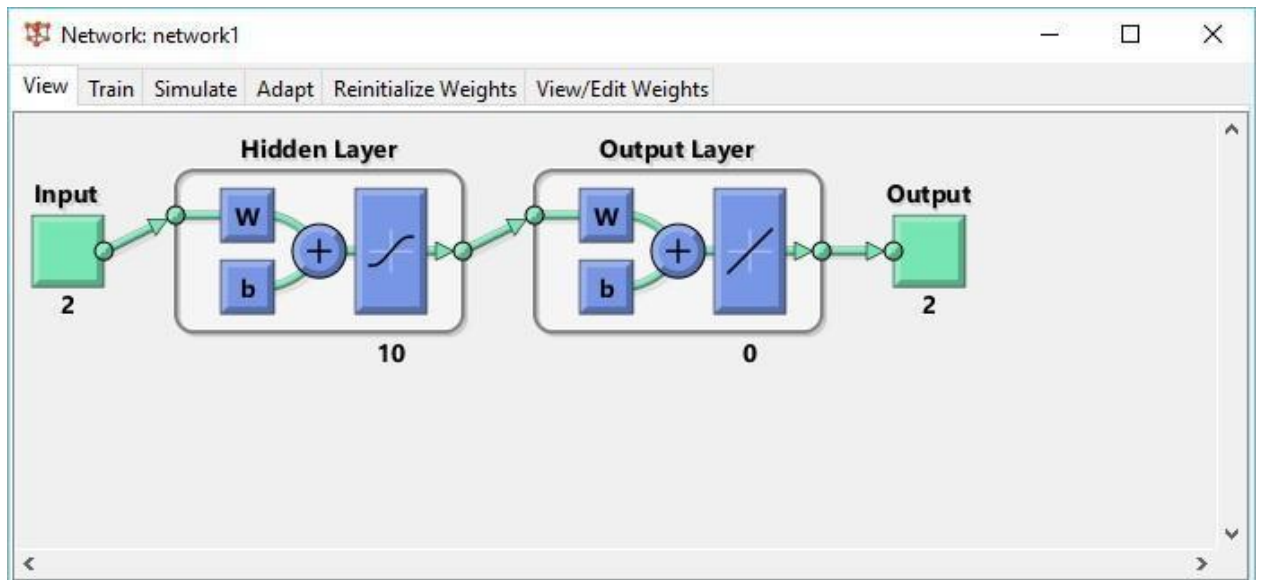


Рисунок 7 □ Диалоговая панель Network

Она открывается только в том случае, когда в окне Network/Data Manager выделена созданная сеть и становятся активными кнопки View,

Initialize, Simulate, Train, Adapt. Панель имеет 6 закладок:

- View (Просмотреть) - структура сети;
- Initialize (Инициализация) - задание начальных весов и смещений;
- Simulate (Моделирование) - моделирование сети;
- Train (Обучение) - обучение сети;
- Adapt (Адаптация) - адаптация и настройка параметров сети; □ Weights (Весы) - просмотр установленных весов и смещений.

Особенности работы с соответствующими окнами будут рассмотрены на приведенных ниже примерах создания конкретных нейронных сетей.

2. Создание программных моделей нейронных сетей Создать и обучить нейронную сеть выполнению операции $y = x_1^2 + x_2$,

если заданы последовательности входа

$$P = [1 \ 0.5 \ 0 \ 1, -2 \ 0 \ 0.5 \ 1]$$

и цели

$$T = [-1 \ 0.25 \ 0.5 \ 2].$$

Сформируем последовательности входов и целей в рабочей области GUI-интерфейса NNTool, используя окно Create New Data.

Выберем нейронную сеть типа feed-forward backprop с прямой передачей сигнала и с обратным распространением ошибки. Схема этой сети показана на рисунке 7.

Затем выполняется обучение сети, для чего выбирается закладка Train и открывается диалоговая панель, показанная на рисунке 8.

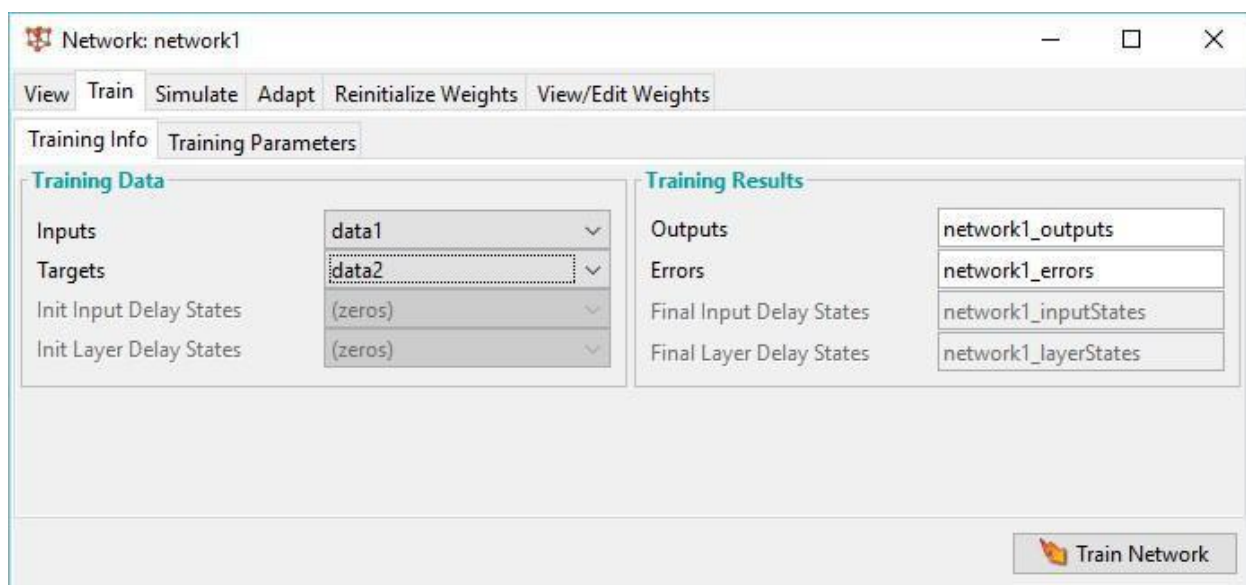


Рисунок 8 □ Диалоговая панель обучения нейронной сети

Панель имеет три закладки:

- **Training Info** (Информация об обучающих последовательностях) (рисунок 8);
- **Training Parametrs** (Параметры обучения) (рисунок 9),

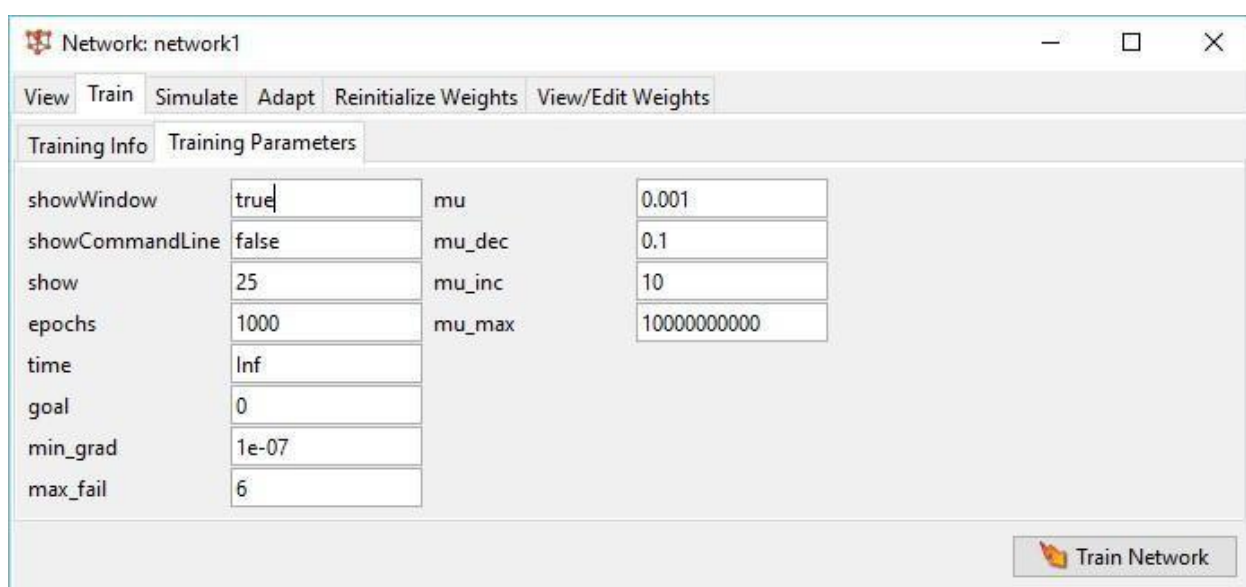


Рисунок 9 □ Диалоговая панель параметров обучения нейронной сети

Применяя эти закладки, можно установить имена последовательностей входа и цели, а также параметров процедуры обучения

Теперь можно приступить к обучению сети (кнопка Train Network).

Качество обучения сети с прямой передачей сигнала на выбранной обучающей последовательности поясняется на рисунке 10. Практически нулевая точность достигается за 13 циклов обучения.

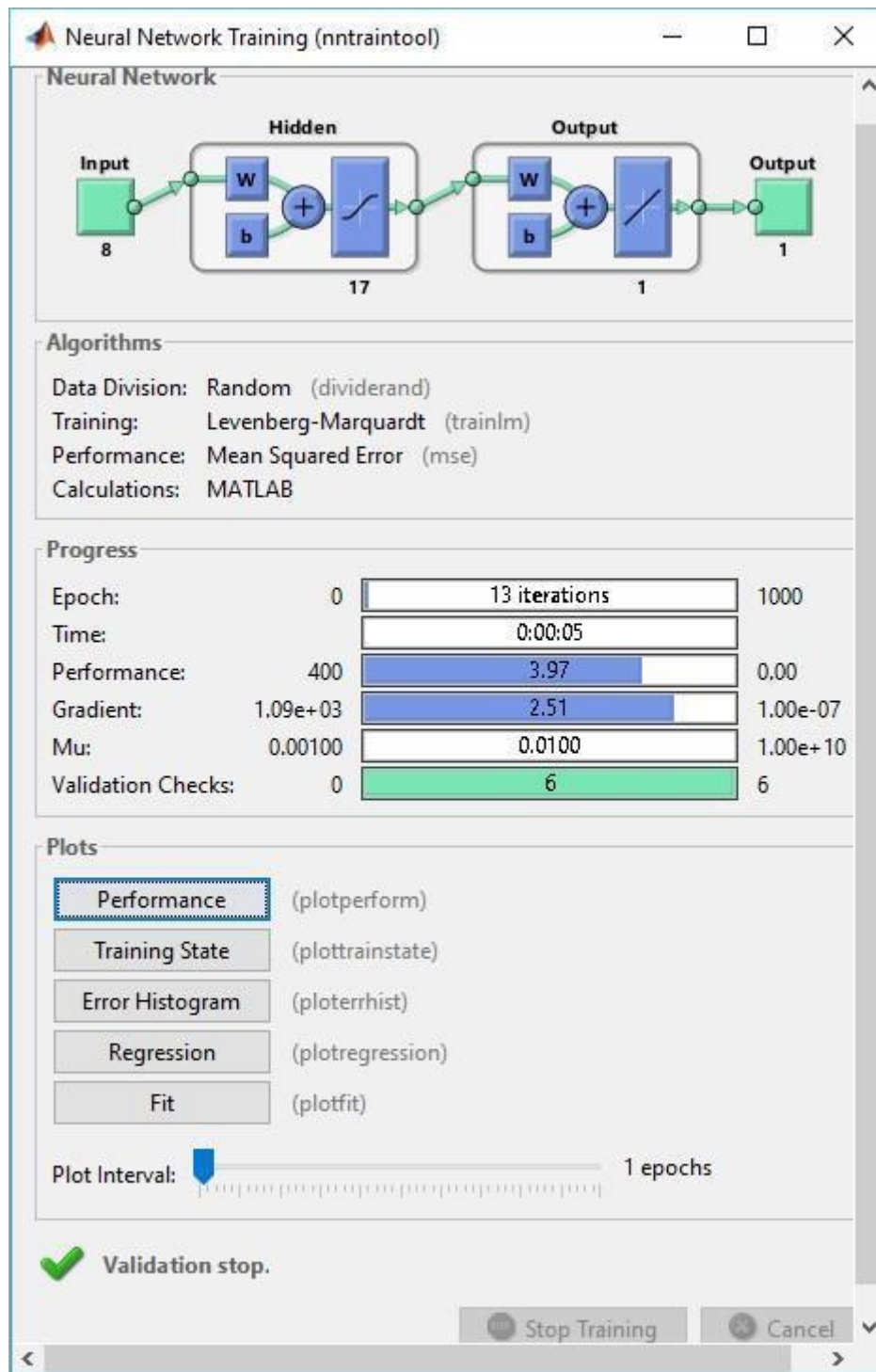


Рисунок 10 □ Процесс обучения нейронной сети

Соответствующие веса и смещения можно увидеть, если выбрать закладку Weights (рисунок 11).

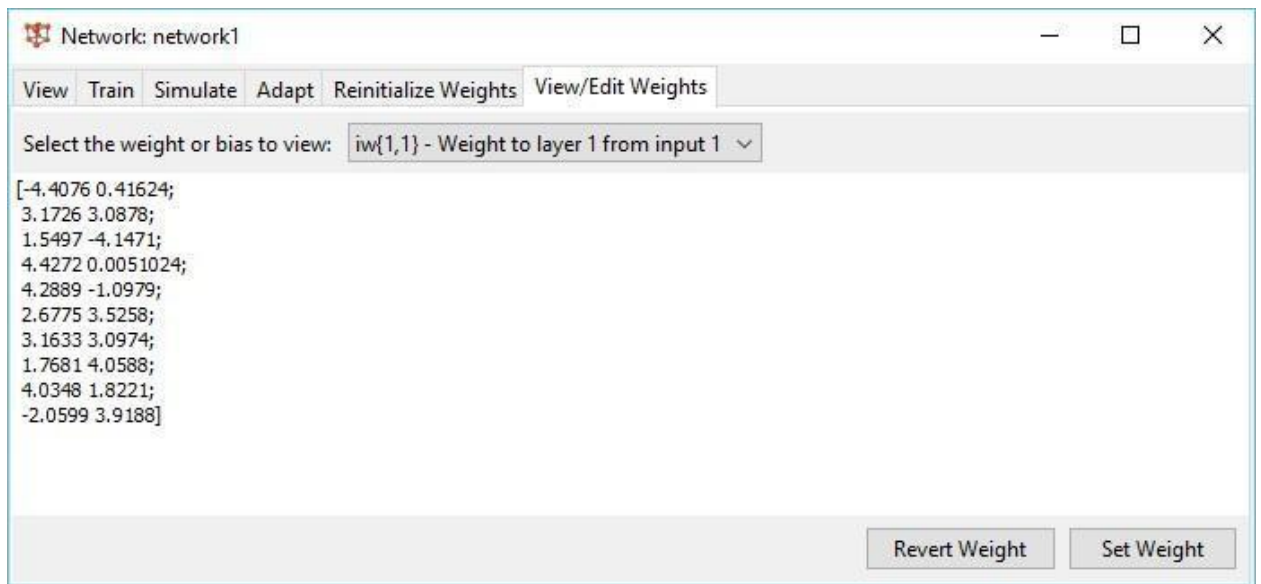


Рисунок 11 □ Панель весов и смещений нейронной сети

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

При помощи пакета Neural Network ToolBox создайте модель нейронной сети прямого распространения с обратным распространением ошибки со следующими параметрами:

Количество входов – 2;

Количество выходов – 1;

Количество нейронов в скрытом слое – 5; Функция активации:

- для скрытого слоя – арктангенс или логистическая; - для выходного – линейная.

Обучите нейронную сеть на основе обучающего примера, приведенного в описании лабораторной работе. Экспортируйте полученную модель в Simulink. Вычертите ее в отчет. Симулируйте нейронную сеть произвольными данными. Сравните полученный выход сети с посчитанным по формуле. Сделайте вывод.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие компьютерные модели нейронных сетей можно создать при помощи пакета Neural Network ToolBox?
2. В чем отличие инструментов моделирования нейронных сетей пакета Neural Network ToolBox версии Matlab 6 и 7?

ПРАКТИЧЕСКАЯ РАБОТА 3. Пакет моделирования нейронных сетей Neural Fitting

Цель занятия:

1. Расширить и углубить знания, полученные на лекциях по основам теории нейронных сетей.
2. Приобрести первичные навыки по проведению исследований и экспериментов с использованием компьютерных моделей информационных систем.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ Инструментальные средства моделирования искусственных нейронных сетей

Отличительной особенностью 7-й версии программы Matlab является расширенный состав инструментальных средств моделирования искусственных нейронных сетей Neural Network Toolbox. Для запуска одного из инструментальных средств пакета в окне команд необходимо набрать:

```
>> nnstart
```

при этом появиться окно, представленное на рисунке 1.

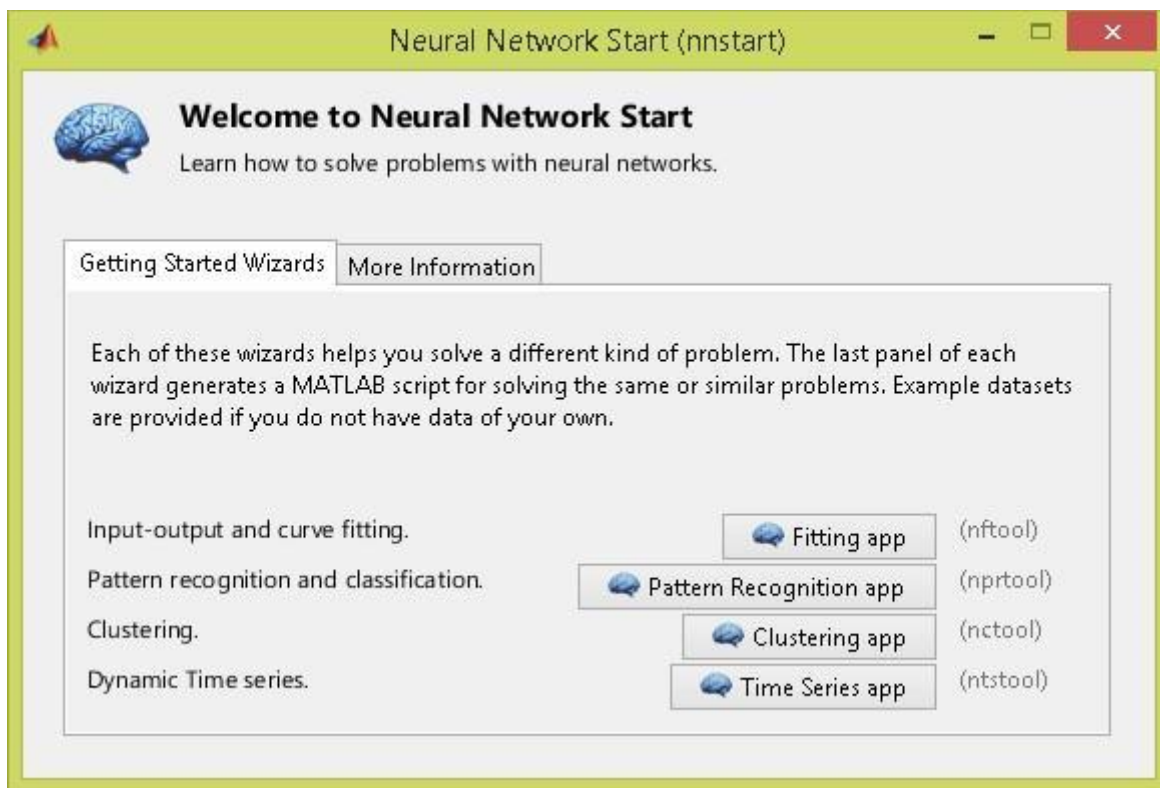


Рисунок 1 – Окно запуска инструментальных средств пакета Neural Network Toolbox

Состав пакета инструментальных средств Neural Network Toolbox приведен в таблице 1.

Таблица 1 – Состав инструментальных средств пакета Neural Network Toolbox

Название	Область применения	Команда
		запуска
Fitting app	Аппроксимация функций	nftool
Pattern recognition app	Распознавание образов	nprtool
Clustering app	Классификация и кластеризация данных	nctool
Time series app	Анализ временных рядов	ntstool

Этапы проектирования компьютерных моделей искусственных нейронных сетей

Процесс проектирования компьютерных моделей искусственных нейронных сетей средствами MATLAB при решении задач приближения функций, распознавания образов, кластеризации и анализа временных рядов можно разбить на следующие этапы:

1. Подготовка исходных данных.
2. Задание архитектура используемой парадигмы нейронной сети.
3. Инициализация весов и смещений.
4. Обучение сети.
5. Проверка функционирования сети.
6. Использование сети.

Постановка задачи приближения функций

Задача аппроксимации функции заключается в определении по экспериментальным данным функции, наилучшим образом приближающейся к неизвестной и удовлетворяющей определенным критериям. Эта задача актуальна при моделировании сложных динамических систем.

Интерфейс пакета Network Fitting

Искусственные нейронные сети являются наилучшим аппроксиматором функций. Воспользоваться инструментальными средствами моделирования искусственных нейронных применительно к задаче приближения функций можно двумя способами:

1. С использованием визуального средства моделирования, запустив приложение Network Fitting Tool при помощи кнопки Fitting app, или командой nftool из окна команд. При этом запустится мастер создания компьютерной модели нейронной сети приложения функции. Окно мастера Neural Fitting представлено на рисунке 2.
2. При помощи интерфейса командной строки, с использование функций пакета Network Fitting Tool.

На левой стороне приложения мастера Network Fitting приведено описание задачи приближения функции, а на правой – изображено и приведено описание архитектуры используемой нейросетевой парадигмы:

1. Нейронная сеть прямого распространения с обратным распространением ошибки.

2. Количество слоев сети – 2.
3. Функция активации нейронов скрытого слоя – логистическая, выходного – линейная.

Для создания компьютерной модели нейронной сети приближения функции необходимо нажать кнопку далее мастера Network Fitting.

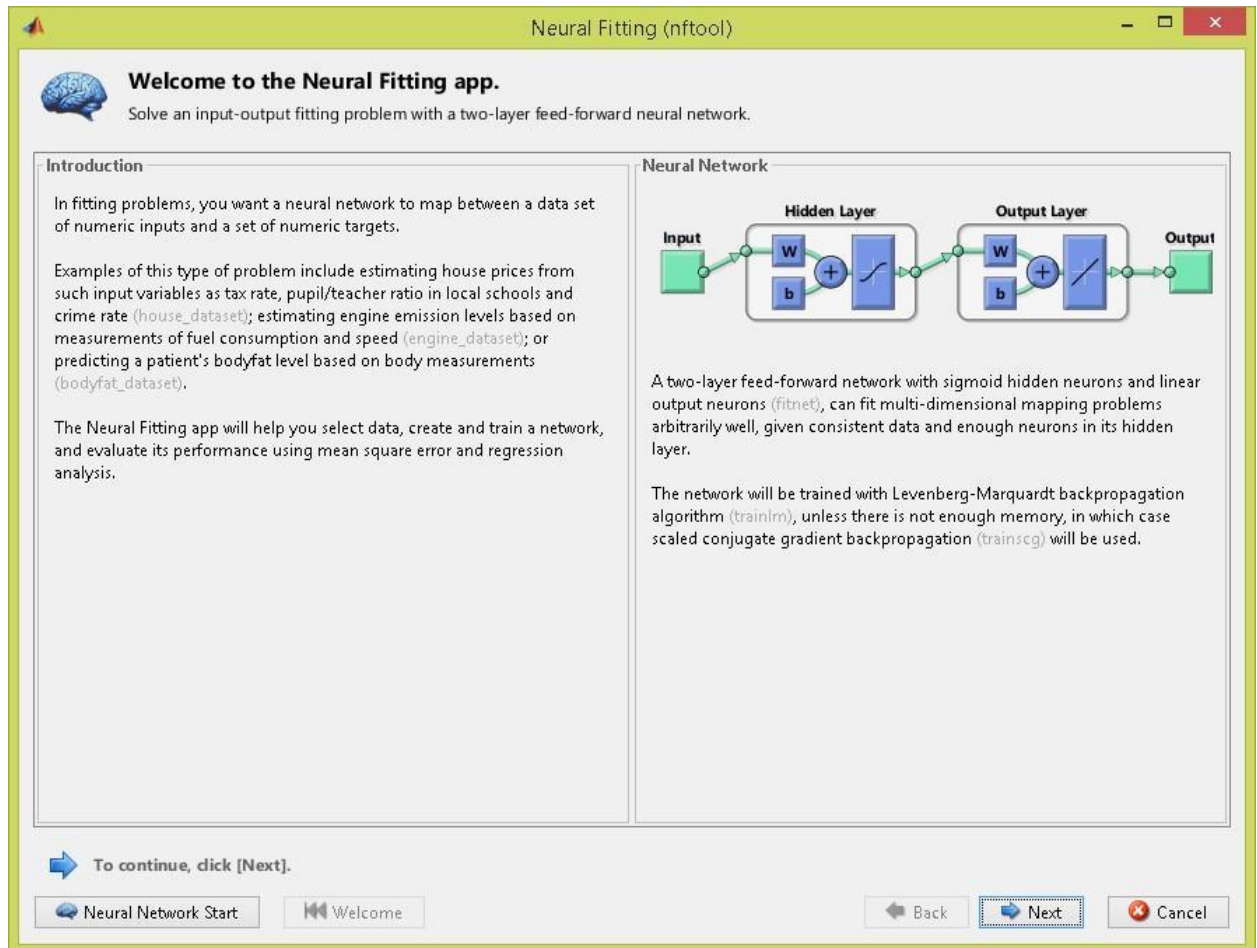


Рисунок 2 – Мастер создания нейронной сети приближения функции

При этом появиться окно второго шага мастера Network Fitting – выбор данных (Select Data), представленное на рисунке 3.

Для примера рассмотрим процесс создания компьютерной модели нейронной сети логической функции И двух аргументов. Для задания исходных данных в рабочей области MATLAB необходимо сформировать матрицу входных данных `inputs`, содержащей все возможные комбинации аргументов и вектор значений логической функции `targets`. В окне команд для этого необходимо набрать:

```
>> inputs = [0 1 0 1; 0 0 1 1];
```

```
>> targets = [0 0 0 1];
```

В рабочей области будут сформированы две матрицы `inputs` и `targets` (рисунок 4).

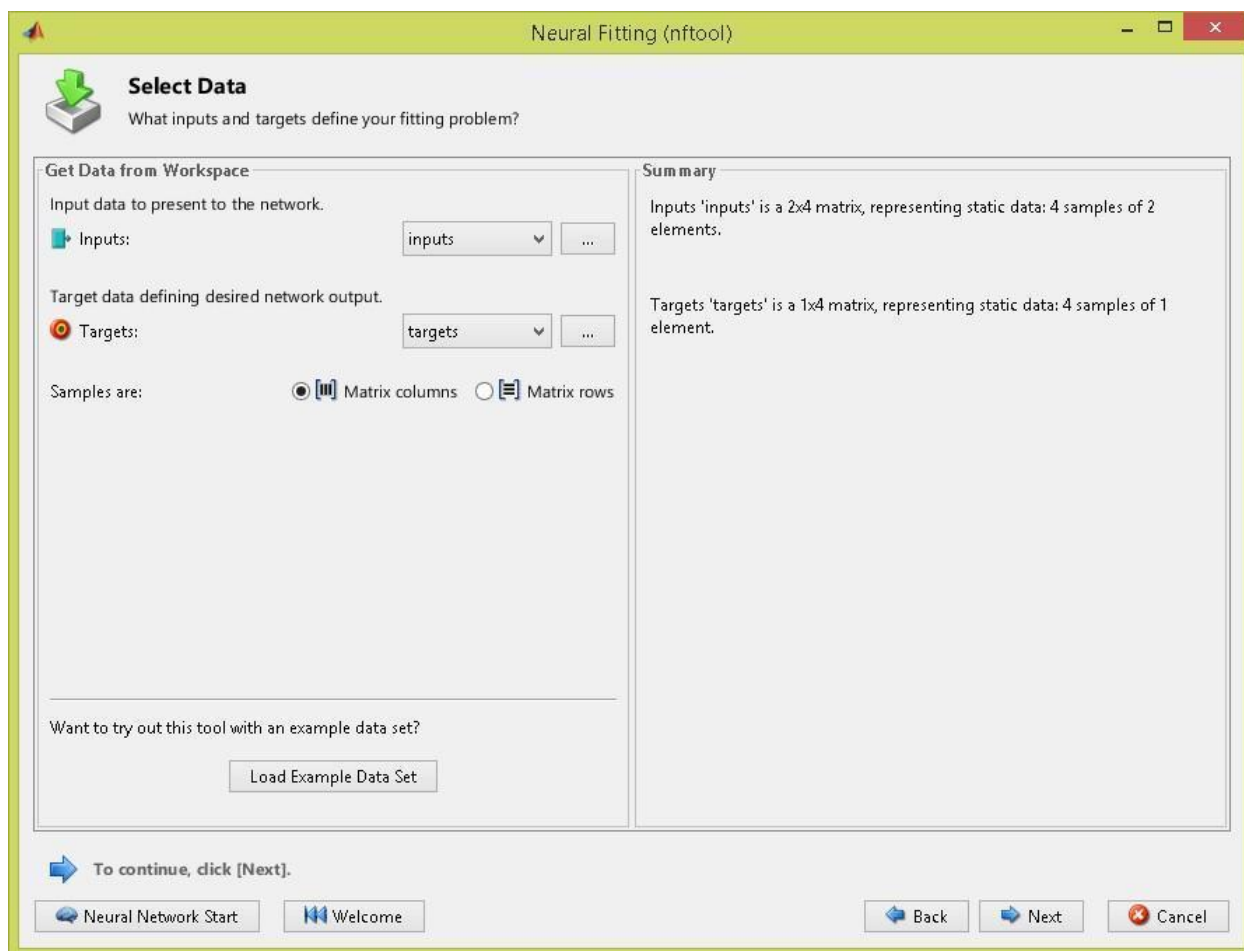


Рисунок 3 – Окно выбора данных мастера создания нейронной сети приближения функции

Workspace	
Name	Value
inputs	[0 1 0 1; 0 0 1 1]
targets	[0 0 0 1]

Рисунок 4 – Рабочая область MATLAB с исходными данными для нейронной сети приближения функции

Для задания исходных данных для обучения нейронной сети в окне выбора данных необходимо выбрать созданные матрицы для входных данных Inputs и выходных данных Targets создаваемой сети. При этом на правой стороне мастера будет сформирован отчет и о проделанной работе. Для продолжения создания нейронной сети необходимо нажать кнопку Next. При этом будет выведено окно третьего этапа мастера создания проверочных и тестовых данных (Validation and Test Data, рисунок 5).

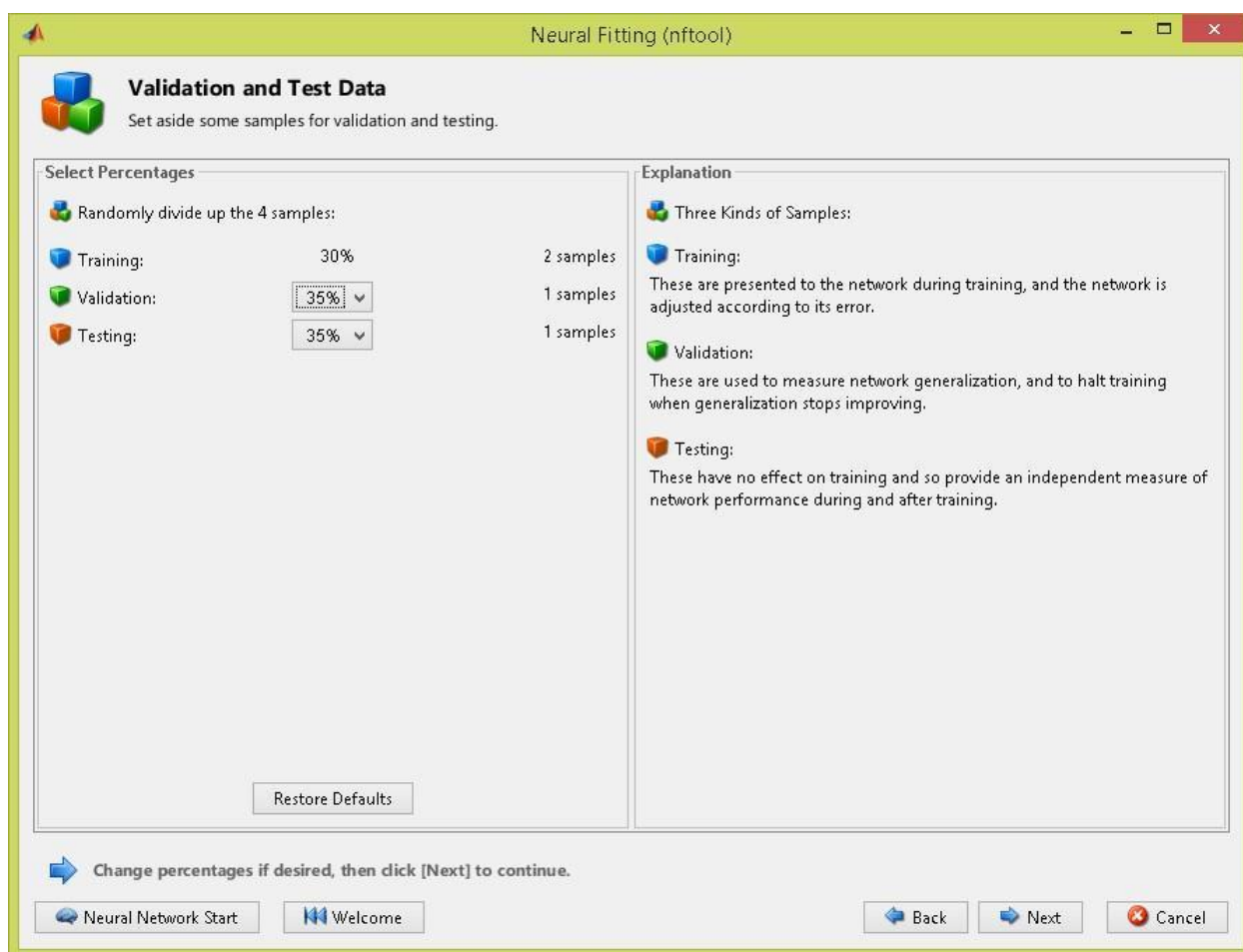


Рисунок 5 – Окно создания проверочных и тестовых данных

На нем представлено распределение исходных данных для обучения нейронной сети на обучающую выборку, тестовые и проверочные данные. Данные этих наборов выбираются случайным образом. При этом имеется возможность увеличить или уменьшить количество наборов для проверочной и тестовой выборки. Так для рассматриваемого примера 30% - обучающая выборка и по 35% - тестовые и проверочные данные.

После задания данных для обучения, тестов и проверки, для завершения этого этапа необходимо нажать кнопку Next. После чего будет выведено окно задания параметров создаваемой нейронной сети Network Architecture (рисунок 6). В поле количество нейронов для скрытого слоя (Number of Hidden Neurons) необходимо ввести требуемое число. По умолчанию мастер предполагает использовать 10 нейронов. Для рассматриваемого примера, при двух входах нейронной сети, достаточно 5 нейронов.

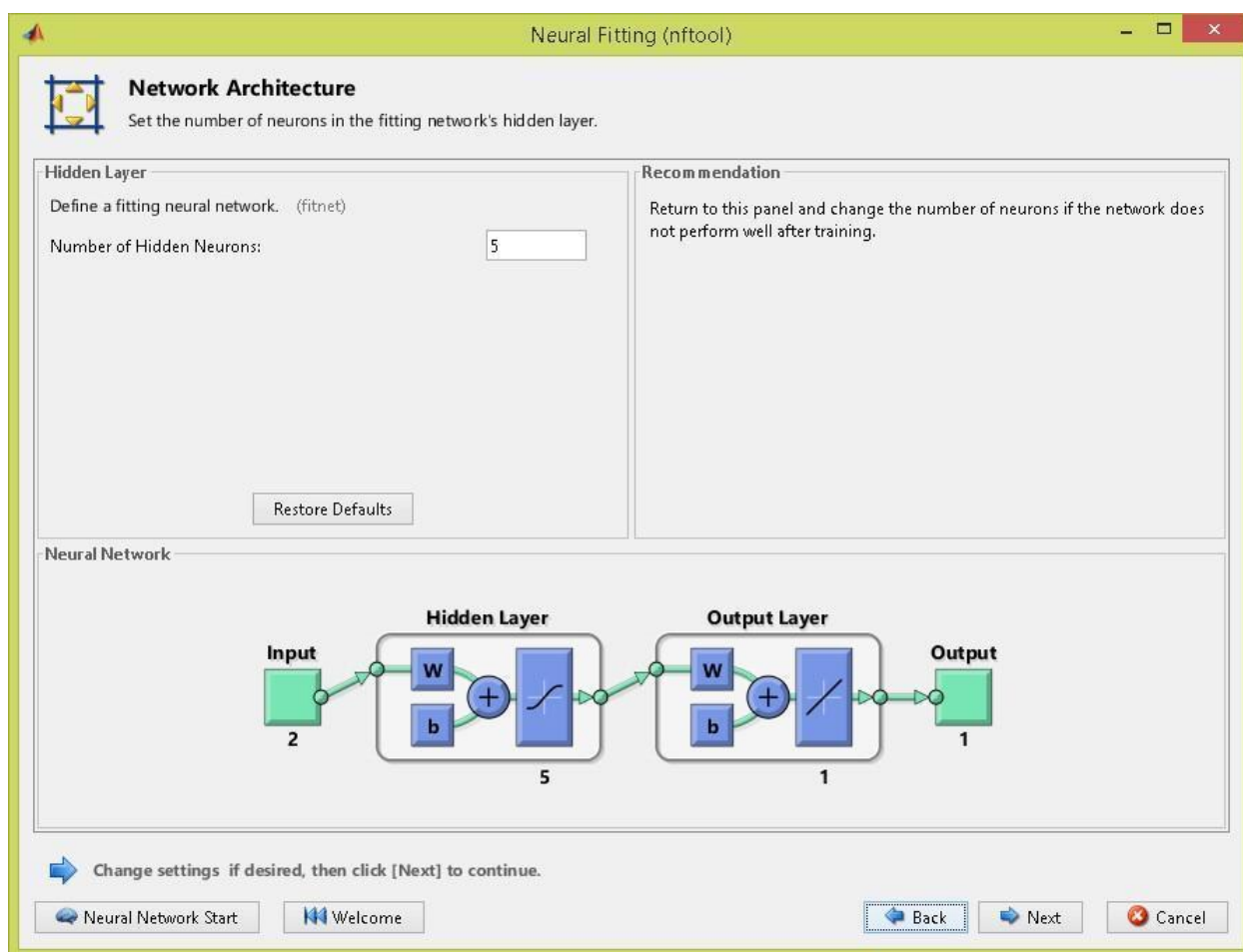


Рисунок 6 – Окно задания параметров архитектуры нейронной сети

После нажатия кнопки Next будет выведено окно мастера, посвященное обучению нейронной сети (Train Network, рисунок 7). Оно позволяет выбрать алгоритм обучения нейронной сети.

В поле выбора алгоритма обучения нейронной сети (Choose a Trading Algorithm, рисунок 7) выбран градиентный алгоритм обучения. Для начала процесса обучения необходимо нажать кнопку Train в правой части окна мастера. После окончания процесса обучения выводится окно (Neural Network Trading, рисунок 8), содержащее результаты обучения нейронной сети. Для рассматриваемого примера нейронная сеть обучилась за 4 итерации до ошибки $7,04 \times 10^{-26}$.

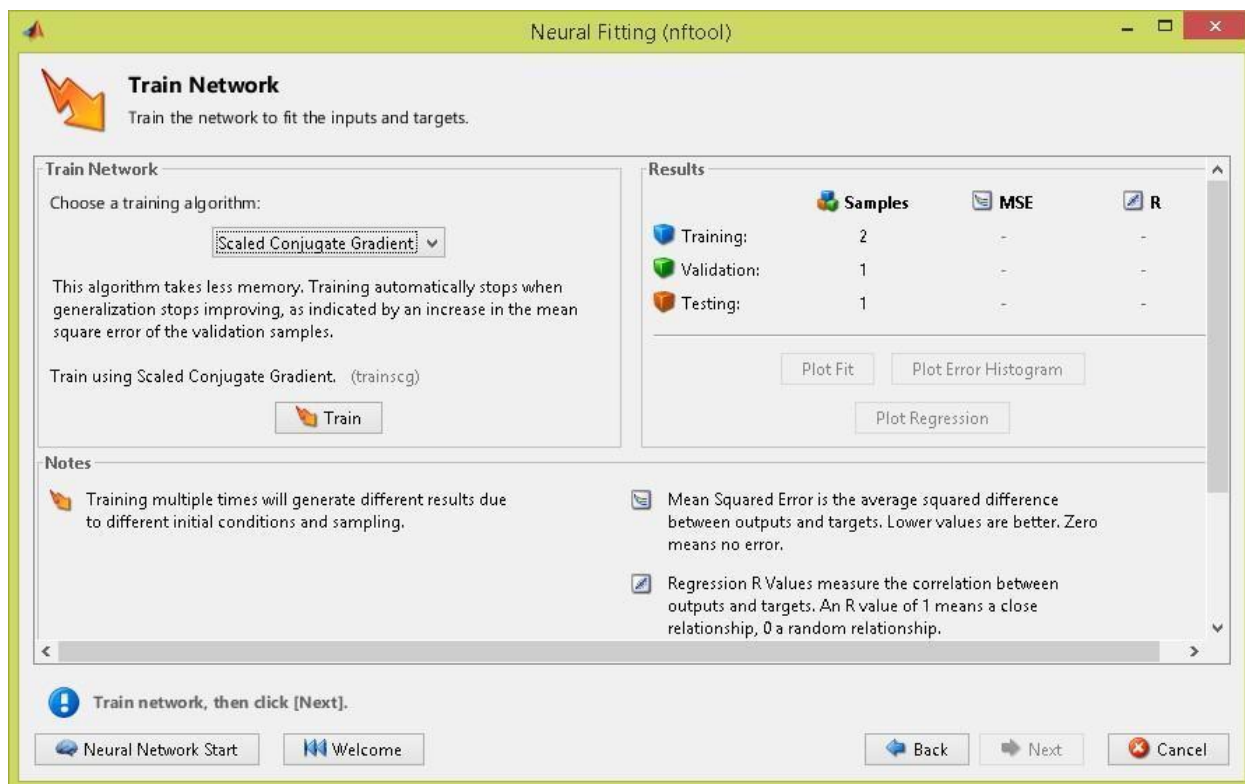


Рисунок 7 – Окно выбора алгоритма обучения нейронной сети

Для построения изменения функции ошибки в процессе обучения необходимо нажать кнопку **Performance** окна **Neural Network Trading**. График изменения функции ошибки представлен на рисунке 9.

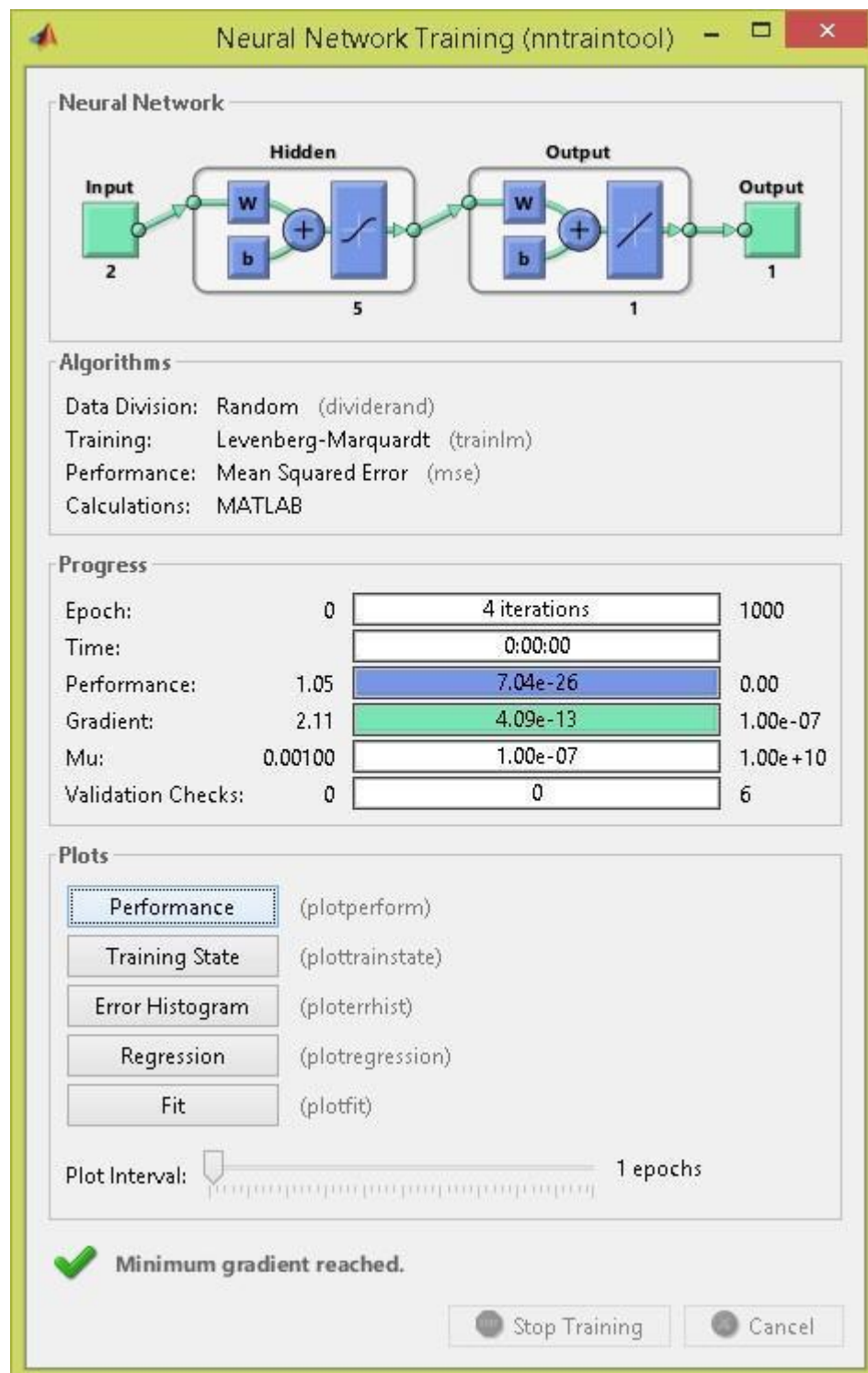


Рисунок 8 – Окно выбора алгоритма обучения нейронной сети

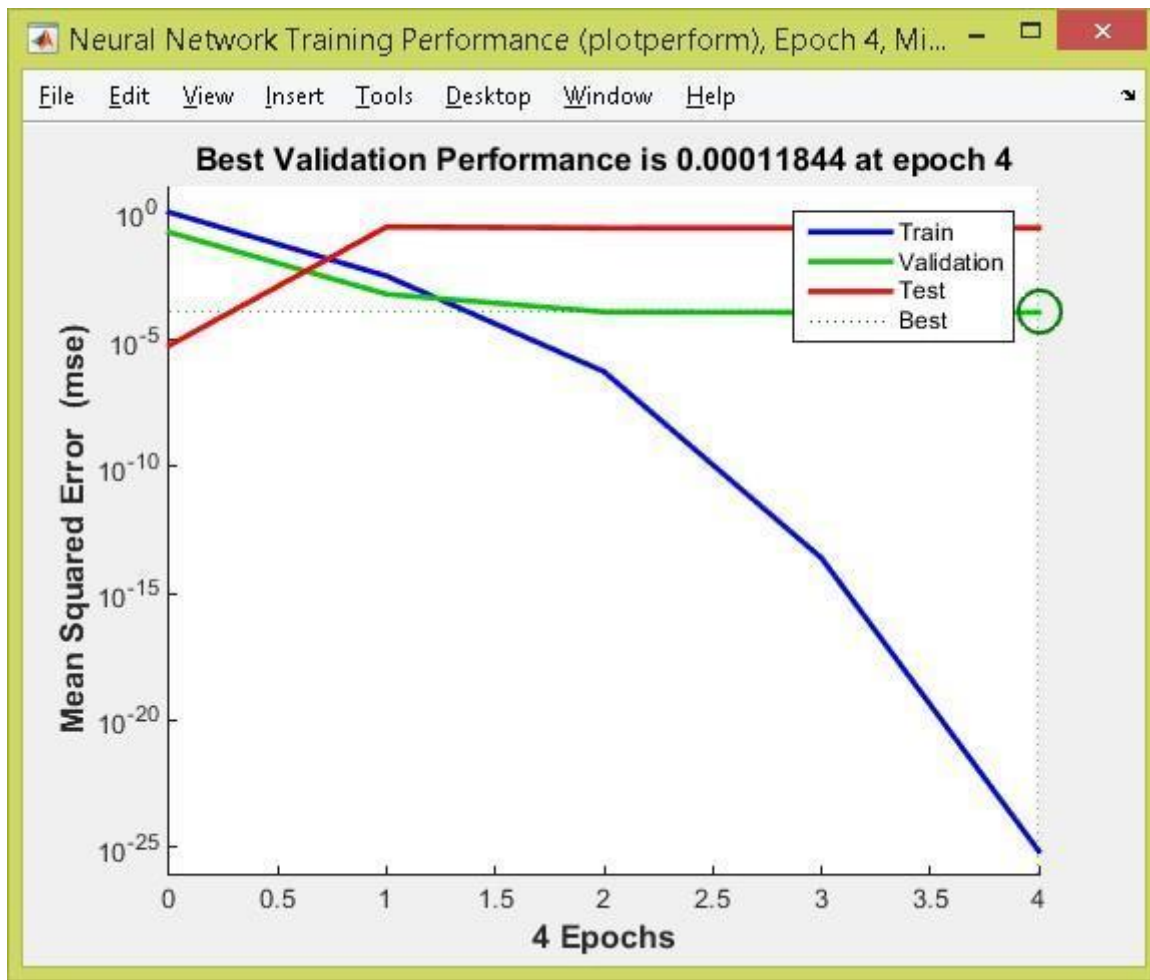


Рисунок 9 – График изменения функции ошибки в процессе обучения нейронной сети

Демонстрационные примеры

Для перехода к демонстрационным примерам пакета Network Fitting на втором шаге работы мастера, при выборе данных Select Data (рисунок 3.2), необходимо нажать кнопку Load Example Data Set. После этого откроется окно Fitting Data Set Chooser (рисунок 3.9), позволяющее загрузить данные для одного из семи демонстрационных примеров:

1. Simple Fitting Problem (пример простейшей аппроксимации),
2. House Pricing (оценка стоимости недвижимости),
3. Abalone Rings (оценка возраста морских моллюсков),
4. Body Fat (оценка лишнего веса для человека),
5. Building Energy (оценка энергетической потребности здания),
6. Chemical (управление химической реакцией),
7. Engine (оценка вращающего момента двигателя).

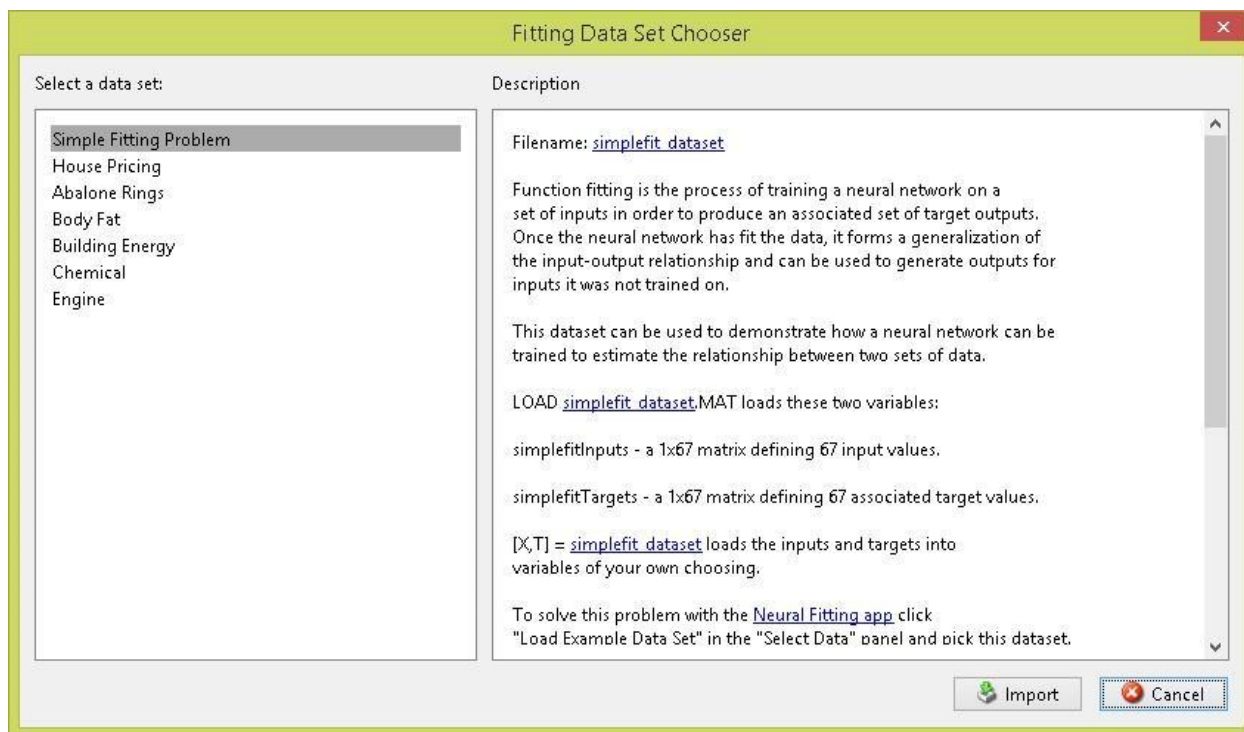


Рисунок 10 – Выбор данных для демонстрационных примеров пакета Network Fitting

Контрольные вопросы

1. Что представляет собой искусственный нейрон?
2. Что называют функцией активации?
3. В чем назначение функции активации?
4. Что называется обучением нейронной сети?

ПРАКТИЧЕСКАЯ РАБОТА 4. Изучение свойств линейного нейрона и линейной нейронной сети

Цель работы: изучить свойства линейного нейрона.

1. Краткие теоретические сведения

Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации `hardlim` с жесткими ограничениями (рис. 2.1).

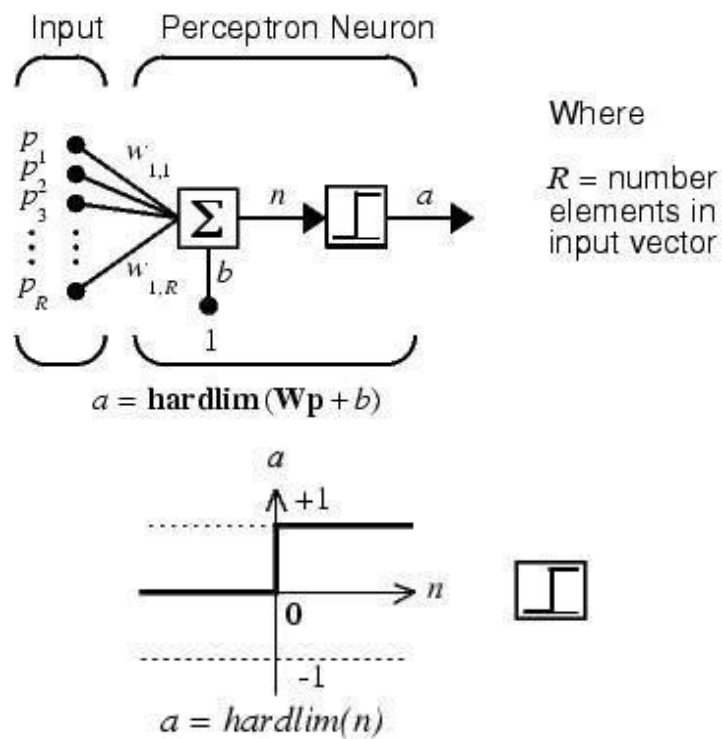


Рисунок 2.1. Структура порогового нейрона

Каждое значение элемента вектора входа персептрона умножено на соответствующий вес w_{ij} , и сумма полученных взвешенных элементов является входом функции активации.

Если вход функции активации $n \geq 0$, то нейрон персептрона возвращает 1, если $n < 0$, то 0.

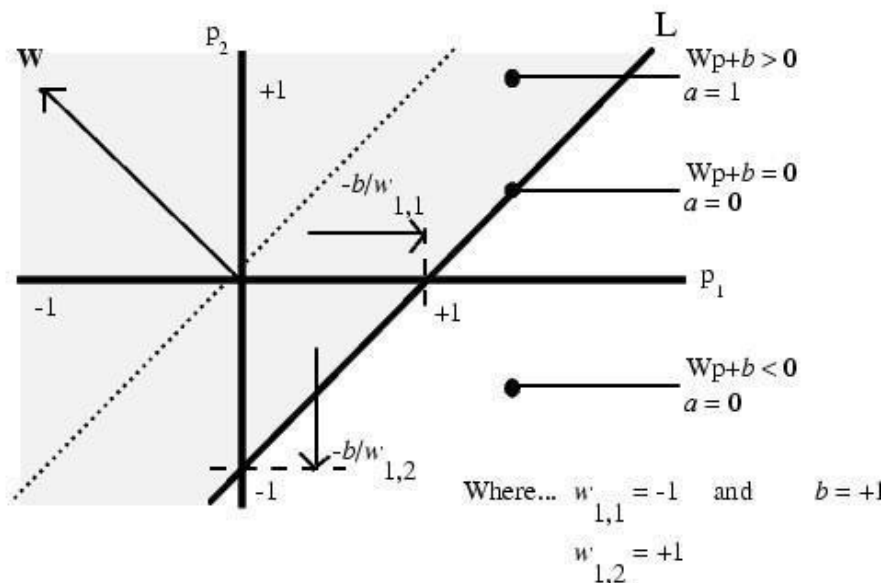


Рисунок 2.2. Разделение нейроном пространства входа на две области

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на две области, как это показано на рисунке 2.2, для персептрона с двумя входами и смещением.

Пространство входов делится на две области разделяющей линией L , которая для двумерного случая задается уравнением

$$w^T p + b = 0. \quad (2.1)$$

Эта линия перпендикулярна к вектору весов w и смещена на величину

б. Векторы входа выше линии L соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1; векторы входа ниже линии L соответствуют выходу персептрона, равному 0. При изменении значений смещения и весов граница линии L изменяет свое положение.

Персептрон без смещения всегда формирует разделяющую линию, проходящую через начало координат; добавление смещения формирует линию, которая не проходит через начало координат, как это показано на рисунке 2.2.

В случае, когда размерность вектора входа превышает 2, т. е. входной вектор p имеет более 2 элементов, разделяющей границей будет служить гиперплоскость.

Для того чтобы создать нейрон, используют функцию *newp*, имеющую следующий синтаксис [56, 57]:

`net = newp(PR, S, TF, LF),`

где PR – матрица минимальных и максимальных R входных элементов, S – количество нейронов (при создании одного нейрона $S = 1$), TF – функция активации (transfer function), LF – имя функции обучения нейрона.

В случае если параметры функции *newp* не заданы, их значения определяются посредством ввода значений в диалоговые окна.

Построенный нейрон характеризуется функциями весов (*weight function*), входов сети (*net input function*) и определенной функцией активации.

Функция весов – это умножение весов на входной сигнал, функция входов сети – их сумма. Веса задаются как для входов нейрона, так и для фиксированного входа, задающего порог срабатывания (*bias*). Вектор весов инициализируется нулями. Для обучения используются функции, рассмотренные ниже.

Функция *learnp* настраивает веса нейрона. Синтаксис функции обучения довольно сложен:

`[dW, LS] = leamp(W, P, Z, N, A, T, E, gW, gA, D, LP, LS),`

`[db, LS] = leamp(b, ones(l,Q), Z, N, A, T, E, gW, gA, D, LP, LS),`

`Info = learnp(code)`

Функция *learnp* ($W, P, Z, N, A, T, E, gW, gA, D, LP, LS$) имеет несколько входов, где вектор W – вектор весов; P – вектор входов; Z – вектор взвешенных входов; N – вектор

сети; A – вектор выхода; T – вектор желаемых выходов; E – вектор ошибок; gW – вектор изменения весов; gA – изменения выходов. Функция возвращает значения: dW – изменения матрицы весов; LS – новый уровень обученности.

Функция `learnp` может быть использована с параметрами по умолчанию: $dW = \text{learnp}([], p, [], [], [], e, [], [], [], [])$.

Использование пустого списка `[]` означает параметр по умолчанию.

Функция `learnp` вычисляет изменение весов dW для заданного нейрона в соответствии с правилом обучения персептрона:

$$dw = \begin{cases} 0, & \text{если ошибка } e = 0, \\ p', & \text{если ошибка } e = 1, \\ -p', & \text{если ошибка } e = -1, \end{cases}$$

$$\text{т.е. } dw = e \cdot p'$$

Тогда новый вектор весов примет вид: $w = w + dw$.

Функция обучает нормализованные веса:

$[dW, LS] = \text{learnpn}\{W, P, Z, N, A, T, E, gW, gA, D, LP, LS\}$. Функция `learnpn` вычисляет изменение весов dW для данного нейрона и его входа P и ошибки E в соответствии с нормализованным правилом обучения персептрона:

$$pn = \sqrt{(1 + p(1)^2 + \dots + p(R)^2)},$$

$$dw = \begin{cases} 0, & \text{если ошибка } e = 0, \\ np', & \text{если ошибка } e = 1, \\ -np', & \text{если ошибка } e = -1, \end{cases}$$

$$\text{т.е. } dw = e \cdot np'$$

Линейный нейрон имеет одно существенное ограничение. Входные векторы должны быть линейно сепарабельны. Если векторы невозможно отделить прямой или гиперплоскостью, то персептрон не способен решить задачу классификации.

Функция `adapt` адаптирует нейрон к условиям задачи:

$$[\text{net}, Y, E, Pf, Af] = \text{adapt}(\text{net}, P, T, P_i, A_i).$$

Параметры функции: `net` – имя сети, `P` – входы сети, `T` – желаемый выход, `Pi` – исходные условия задержки, `Ai` – исходные условия задержки для слоя. Функция возвращает параметры адаптированной сети `net.adaptParam`: `net` – измененная сеть, `Y` – выход сети, `E` – ошибки сети, `Pf` – условия задержки входов, `Af` – условия задержки слоя. Параметры `Pi` и `Pf` необязательные и необходимы только для сетей, имеющих задержки на входах и слое.

Функция `train` также обучает нейронную сеть и использует следующий синтаксис:

$$[\text{net}, \text{tr}] = \text{train}(\text{NET}, P, T, P_i, A_i)$$

$$[\text{net}, \text{tr}] = \text{train}(\text{NET}, P, T, P_i, A_i, W, TV).$$

Функция $\text{train}(\text{net}, P, T, P_i, A_i)$ имеет следующие параметры: net – сеть, P – входы сети, T – желаемый выход, P_i – исходные условия задержки входа, A_i – исходные условия задержки слоя.

Функция эмулирует нейронную сеть:

$[Y, Pf, Af] = \text{sim}(\text{net}, P, P_i,$

$A_i)$, где net – сеть;

P – входы сети;

P_i – исходные условия задержки входов сети;

A_i – исходные условия задержки слоя.

Функция возвращает

Y – выходы сети;

Pf – окончательные условия задержки входов;

Af – окончательные условия задержки слоя.

Ниже представлены назначения этих функций.

Таблица 2.1. Функции активации

Функция	Назначение
$\text{hardlim}(N)$	Возвращает 1, если N положительное и 0 в противном случае
$\text{tansig}(N)$	Вычисляет гиперболический тангенс от входа
purelin	Вычисляет выход слоя от сетевого входа

Таблица 2.2. Функции графического интерфейса и вспомогательные функции

Функция	Назначение
$\text{axis}([Xmin\ Xmax\ Ymin\ Ymax])$	Устанавливает диапазоны координатных осей
$\text{title} \{ 'строка' \}$	Выводит в графическое окно рисунков заголовок графика
$\text{rand}(M, N)$	Возвращает матрицу размерности M на N со случайными значениями
$\text{xlabel} \{ 'строка' \} \text{ ylabel} \{ 'строка' \}$	Подписывают наименование координатных осей
cla reset	Очищает координатную сетку в окне рисунков

hold on и hold off	Включают и отключают режим добавления графиков на координатную сетку
text(X, Y, 'строка')	Выводит строку, начиная с указанных координат в поле рисунков
pause (n)	Ожидает пользовательского ответа n секунд
plot(X, Y, 'цвет и символ')	Изображает на координатной сетке точки с координатами, заданными векторами X, Y, с помощью указанного символа и цвета ¹ .
Функция	Назначение
plotpv(P, T)	Изображает точки P указанными маркерами T, где P – матрица входных векторов размерностью R на Q (R должен быть 3 или меньше), T– матрица двоичных векторов размерностью S на Q (S должен быть 3 или меньше)
plotes(WV, BV, ES,V)	Изображает поверхность ошибки на отдельном входе, где WV– вектор строк значений весов W размерности N, BV– вектор строк значений порогов B размерности M, ES – матрица ошибки
	размерности M на N, V– угол зрения по умолчанию [-37,5, 30]
plotsom(POS)	Изображает позицию нейрона красной точкой, связывая синей линией нейроны, находящиеся друг от друга на расстоянии 1. POS – матрица S N-размерных нейронов
ind2vec и vec2ind	Позволяют представить индексы либо собственно значениями индексов, либо векторами, строки которых содержат 1 в позиции индекса ²
full	Преобразует разреженную матрицу в полную
maxlinlr(P)	Функция возвращает максимальный уровень обученности линейного слоя без bias, который обучался только на векторе P
trainlm	Выполняет обучение многослойной НС методом Левенберга– Марквардта

netprod	Входная сетевая функция, которая вычисляет выход сетевого слоя, умножая входной вектор на веса и прибавляя bias
init	Итеративно инициализирует НС

¹ plot{X, Y, 'g+:') изображает точки зеленого цвета с помощью символа «+».

² для четырех векторов (содержащих только одну 1 в каждой строке) можно найти индексы единиц: vec = [1 0 0 0; 0 0 1 0; 0 1 0 1]; ind = vec2ind(vec).

Структура данных описания нейронных сетей. Структура данных net – это описание обученной НС. Обучение осуществляется в соответствии со следующими параметрами, значения которых либо устанавливаются пользователем, либо по умолчанию (табл. 2.3).

Таблица 2.3. Структура данных описания нейронных сетей

Структура данных	Комментарий
net.trainParam.epochs 100	максимальное количество эпох обучения
net.trainParam.goal 0	целевое значение ошибки
net.trainParam.max_fail 5	максимальное значение ошибки
net.trainParam.mem_reduc 1	фактор оптимизации процесса обучения: оптимизация использования памяти или времени процессора
net.trainParam.min_grad 1e-10	минимальное значение градиента
net.trainParam.show 25	количество эпох между показами
net.trainParam.time inf	максимальное время обучения в секундах
TR	структура данных, содержащая значения об обученности НС в текущую эпоху
TR.epoch	номер эпохи
TR.perf	уровень обученности (Trainingperformance)
TR.vperf	степень качества (Validationperformance)
TR.tperf	результативность обработки теста (Testperformance)
TR.mu	значение адаптивности

Структура данных описания адаптированной НС net.adaptfcn включает в себя следующие поля net.adapt.param:

NET – адаптированная НС;

Y – выходы НС;

E – ошибки НС;

Pf – окончательные входные значения задержек;

Af – окончательные выходные задержки;

TR – результат обучения (эпохи и целевая ошибка).

Проведем в среде MatLab toolbox эксперименты, используя рассмотренные функции.

2. Создание нейрона, выполняющего функцию логического И

Создадим нейрон с одним двухэлементным входом (интервалы первого и второго элементов [0; 1]). Определим два первых параметра функции *newp*, а в качестве значений третьего и четвертого параметра (типа функции активации и имени процедуры обучения нейрона) воспользуемся значениями по умолчанию.

```
net = newp([0 1; 0 1], 1); % создание нейрона с одним двухэлементным  
входом (интервал первого элемента [0; 1] и второго элемента [0; 1]).
```

Для того чтобы исследовать поведение нейрона, необходимо имитировать его работу с помощью функции *sim*. Для определения последовательности значений входа создадим последовательность P1.

```
P1 = {[0; 0], [0; 1], [1; 0], [1; 1]}; % создание последовательности значений входа
```

Y = sim(net, P1); % имитация работы нейрона net на последовательности входов P1
желаемых выходов – T, которая позволит нам провести адаптацию нейрона (обучить его)
через 20 проходов.

```
T1 = {0, 0, 0, 1}; % создание последовательности выходов
```

```
net.adaptParam.passes = 20; % установка количества проходов (циклов) адаптации
```

```
net = adapt(net, P1, T1); % адаптация нейрона net для обучающей выборки <P1;T>
```

```
Y = sim(net, P1) % симуляция работы нейрона net на последовательности входов P1
```

```
Y = [0] [0] [0] [1]
```

В результате мы получим нейрон, выполняющий функцию конъюнкции.

3. Обучение нейрона выполнению функции логического ИЛИ

Для переобучения нейрона на выполнение функции ИЛИ

переопределим входы P и выходы T.

```
P2 = [0 0 1 1; 0 1 0 1]; % создание последовательности входов
```

T2 = [0, 1, 1, 1]; % создание последовательности выходов (реакций) для нейрона,
выполняющего функцию логического ИЛИ

Инициализируем нейрон, обучим его на 20 проходах (эпохах).

```
net = init(net); % инициализация нейрона net
```

```
Y = sim(net, P2); % имитация работы нейрона net на последовательности входов P2
```

```
net.trainParam.epochs = 20; % установка количества проходов
```

```
net = train(net, P2, T2); % обучение нейрона net на обучающей выборке <P2, T2>
```

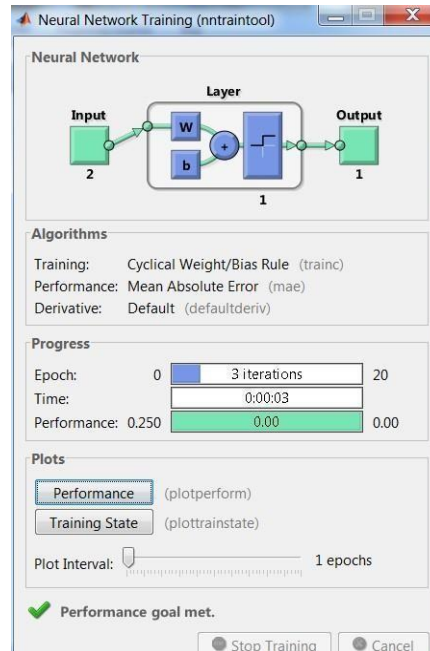


Рисунок 2.3. Окно обучения нейронной сети

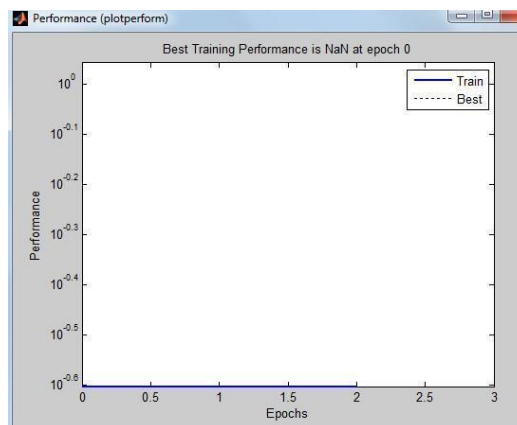


Рисунок 2.4. Изменение ошибки сети в процессе обучения

```
Y = sim(net, P2) % имитация работы нейрона net на последовательности входов P2
```

Y = 0 1 1 1

Для случайного изменения весов и порога срабатывания используем функцию `init`. По умолчанию для создаваемого нейрона указана функция `hardlim`.

4. Определение входов со случайными значениями

Определим случайный вход p и ошибку e для нейрона с двухэлементным входом.

```

p = rand(2,1); % определение входа p как вектора двух случайных
чисел из интервала [0;1]

e = rand(0,1); % определение входа e как случайного числа из
интервала [0;1]

```

5. Построение графика функции активации **hardlim**

График функции активации для интервала $[-3; +3]$ с шагом 0.1.

```

n = -3:0.1:3; % определение переменной n со значениями из интервала [-3; 3] с
шагом 0.1

```

```

b = hardlim(n); % вычисление пороговой функции от переменной n

```

```

plot(n, b); % изображение графика функции  $b = \text{hardlim}(n)$ 

```

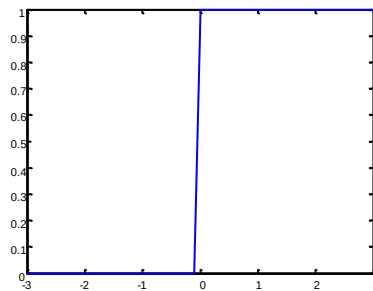


Рисунок 2.5. График функции активации **hardlim**

6. Имитация работы линейного нейрона

Создадим с помощью **newp** нейрон с двухэлементным входом

(значения входа в интервале $[0;1]$):

```

net = newp([0 1;0 1], 1);

```

Функция **sim** имитирует работу нейрона для отдельного вектора и для трех векторов.

```

p1 = [.3; .7]; % определение двухэлементного вектора

```

```

a1 = sim(net, p1); % имитация работы нейрона net на входном векторе
результат имитации – вектор a1

```

```

p2 = [.3 .5 .2; .8 .5 .4]; %определение матрицы входов размерностью 3
на 2 (три двухэлементных вектора)

```

```

a2 = sim(net, p2); % имитация работы нейрона net на входном векторе
p2, результат имитации – вектор a2

```

7. Обучение нейрона классификации векторов на две категории

Начнем с классификации векторов на основе двухвходового нейрона.

Будем использовать функцию *newp* для создания нейрона, *sim* для имитации его работы, *adapt* для адаптации (обучения) нейронной сети. Обучим двухвходовой нейрон классифицировать входные векторы на две категории.

Определим четыре входных вектора нейрона.

P = [-0.5 -0.5 +0.4 -0.2; -0.5 +0.5 -0.4 +1.0]; % определение четырех двухэлементных входов

Зададим желаемые выходы нейрона для определенных векторов.

T = [1 1 0 0]; % определение выходов нейрона – реакций на входы P.

Изобразим входные векторы (точки) на плоскости:

plotpv(P, T)

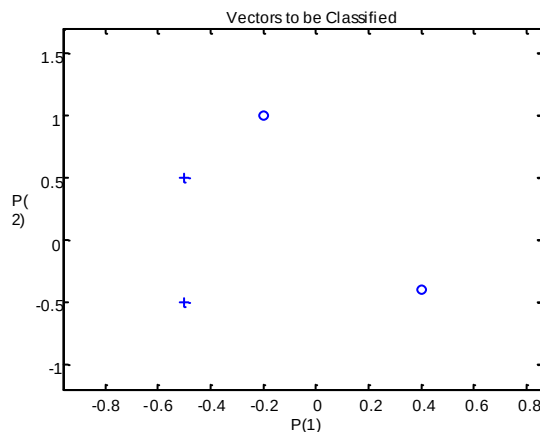


Рисунок 2.6. Изображение входных векторов на плоскости

Каждый из четырех входных векторов на плоскости P определяется двумя координатами, представленными как двухэлементные столбцы в матрице P.

Создадим один линейный нейрон с двумя входами, значения которых лежат в интервале [-1,1].

net = newp([-1 1; -1 1], 1); % создание линейного нейрона с двумя входами из интервала [-1, 1]

Нейрон по умолчанию имеет функцию активации *hardlim* и такой нейрон разделяет входные векторы прямой линией.

Определим координаты линии классификации: веса (IW) и порог срабатывания нейрона (b).

linehandle = plotpc(net.IW{1},net.b{1}); % получение управляющей структуры linehandle для изображения разделяющей линии в координатах весов (IW) и порога срабатывания нейрона (b).

plotpc(net.IW{1}, net.b{1}); % изображение разделяющей прямой

Если исходным весам задать нулевые значения, то любые входы дадут одинаковые выходы, и линия классификации не будет видна на плоскости.

Проведем обучение:

```
clf; % очистка координатных осей
```

```
plotpv(P, T); % изображение входных векторов двух категорий,
```

категория задается элементами вектора T

```
linehandle = plotpc(net.IW{1}, net.b{1}); % получение управляющей
```

структуры linehandle

```
E = 1; % присвоение начального значения ошибки
```

```
net = init(net); % инициализация нейрона
```

```
linehandle = plotpc(net.IW{1}, net.b{1}); % получение управляющей структуры  
linehandle
```

```
while(mse (E)); % организация цикла пока ошибка не равна 0
```

```
[net, Y, E] = adapt(net, P, T); % адаптация нейрона net на обучающей выборке <P,  
T>, функция возвращает адаптированный нейрон net, выход Y, ошибку E
```

```
linehandle = plotpc(net.IW{1}, net.b{1}, linehandle); % изображение
```

разделяющей прямой нейрона после адаптации

```
drawnow; % очистка окна графиков
```

```
end; % конец цикла while
```

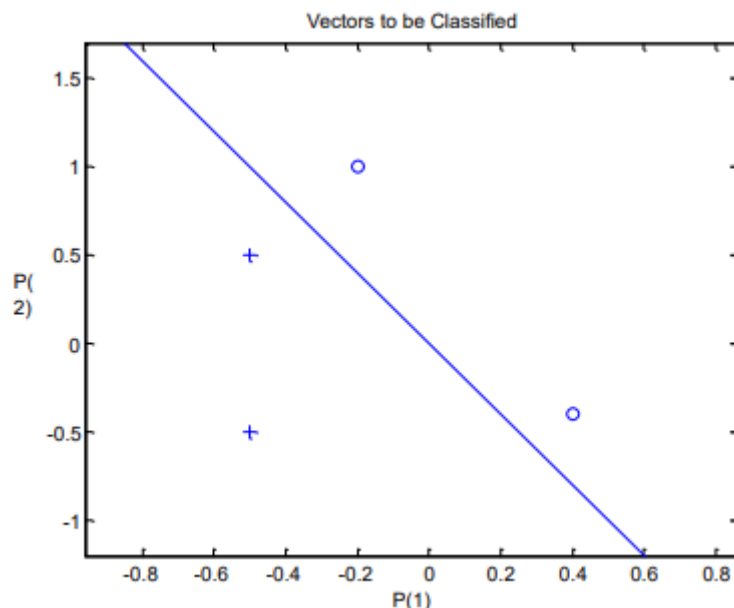


Рисунок 2.7. Изображение разделения пороговым нейроном входных векторов

Функция `adapt` возвращает новый объект – сеть, которая выполняет классификацию, выход сети и ошибку.

Проведем классификацию нового вектора с помощью обученного нейрона на основе функции *sim*. Определим новый вектор *p*.

```
P = [0.6; 1.1]; % определение вектора P
```

```
a = sim(net, P); % имитация работы нейрона net, получение отклика  
нейрона a
```

```
plotpv(P, a); % изображение входа P, отнесенного нейроном к  
категории a
```

Обученный нейрон можно использовать для классификации любого вектора.

```
hold on; % включить режим добавления графиков в графическом окне
```

```
plotpv(P, T); % изображение входных точек в соответствии с  
категориями T
```

```
plotpc(net.IW {1}, net.b{1}); % изображение разделяющей поверхности
```

```
hold off; % отключение режима добавления графиков
```

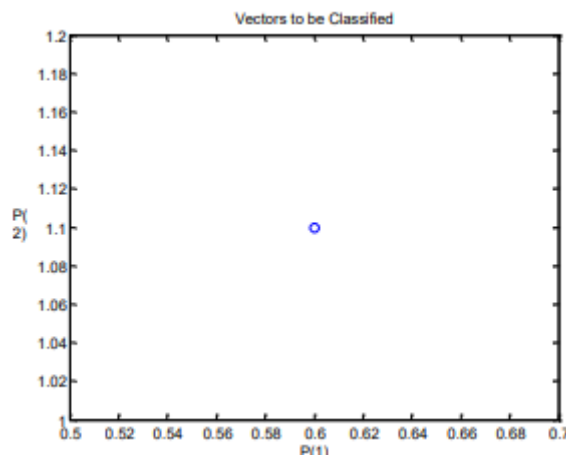


Рисунок 2.8. Изображение входа *P*, отнесенного нейроном к категории *a*

Нейрон классифицирует новую точку, как принадлежащую категории «0» (представлена кружком), а не категории «1» (представлена +).

8. Создание слоя линейных нейронов

Рассмотрим последовательность из 10 шагов (для выхода *T1*, который известным образом зависит от входов *P1*):

```
P1 = [-1 0 0 0 1 1 -1 0 -1 1]; % последовательность входов
```

```
T1 = [-1 -1 1 0 1 2 0 -1 -1 1]; % последовательность выходов
```

Используем функцию *newlin*, чтобы создать нейрон со значениями входа в интервале $[-1;1]$, задержками входа от 0 до 1 и уровнем обучения 0,1.

net = newlin([-1 1], 1, [0 1], 0.1); % создание линейного слоя из одного нейрона со значениями входа в интервале [-1; 1], задержками входа от 0 до 1 и уровнем обучения 0,1.

Адаптируем нейрон к задаче одним проходом через последовательность входа. Измерим среднюю квадратичную ошибку с помощью функции **mse(e)**.

[net, y, e, pf] = adapt(net, P1, T1); % адаптация нейрона к последовательности P1
mse(e) % измерение ошибки ans = 1.1000

Получим довольно большую ошибку. Вновь адаптируем сеть на 10 шагах последовательности, используя предыдущее значение Pf как новое исходное значение задержки:

p2 = [1 -1 -1 1 1 -1 0 0 0 1]; t2 = [2 0 -2 0 2 0 -1 0 0 1];

[net, y, e, pf] = adapt(net, p2, t2, pf); % адаптация с начальным вектором задержки
pf mse(e) ans = 0.6476

Адаптируем сеть на 100-разовом прогоне последовательности:

p3 = [P1 p2]; % формирование новой последовательности

входов

t3 = [T1 t2]; % формирование новой последовательности

выходов **net.adaptParam.passes = 1000;** % установка количества

проходов

[net, y, e] = adapt(net, p2, t2); % адаптация нейрона **mse(e);**

Получим приемлемую ошибку, значит сеть обучена зависимости выходов от входов.

9. Изучение возможности линейного нейрона решать линейно несепарабельные задачи

Проведем эксперимент с линейным нейроном по классификации пяти входных векторов на две категории, которые линейно несепарабельны.

Определим входы линейного нейрона:

P = [-0.5 -0.5 +0.4 -0.1 -0.9; -0.5 +0.5 -0.5 +1.0 +0.0]; % входы линейного нейрона

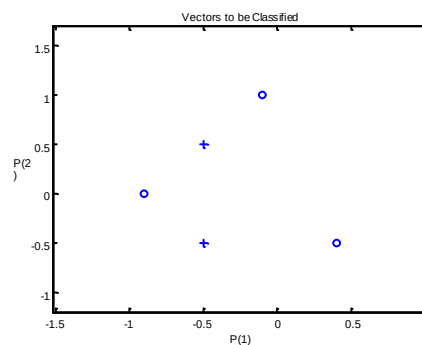


Рисунок 2.9. Изображение входных векторов на плоскости

```
T = [1 1 0 0 0]; % выходы – реакции нейрона на входы P
```

Поскольку векторы разных видов нельзя отделить прямой линией, их классификация невозможна.

```
net = newp([-2 2; -2 2],1); % создание нейрона
```

```
plotpv(P,T); % изображение обучающей выборки
```

```
linehandle = plotpc(net.IW{1}, net.b{1}); % изображение разделяющей  
прямой
```

Обучим нейрон классификации векторов.

```
for a = 1:20; % организация цикла обучения нейрона
```

```
[net, Y, E] = adapt (net, P, T); % адаптация нейрона
```

```
linehandle = plotpc(net.IW{1}, net.b{1}, linehandle); % получение  
управляющей структуры изображения для новой разделяющей прямой
```

```
drawnow; % изображение разделяющей прямой
```

```
end; % конец цикла
```

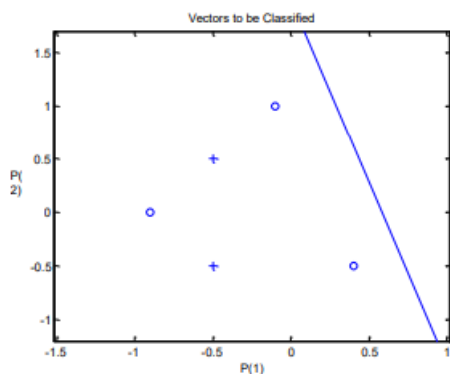


Рисунок 2.10. Изображение невозможности
разделения пороговым нейроном линейно
неразделимых векторов

Цикл дает возможность адаптировать нейрон при одном проходе, изобразить линию классификации и остановиться. Нулевая ошибка, а следовательно, и классификация никогда не будет получена.

Фундаментальное свойство линейного нейрона – ограничение линейно сепарабельными данными.

10. Процесс обучения линейного нейрона

Для выполнения примера будем использовать функции: *newlin* – для создания нейрона, *train* – для обучения, *sim* – для имитации поведения нейрона.

Определим входы нейрона и желаемые выходы: **p = [1.0 -1.4];** % определение входов

нейрона

T = [0.5 1.0]; % определение выходов нейрона

Вектор **p** задает два входных элемента. Используем линейный нейрон с одним входом для решения задачи.

w = -1:0.1:1; % установка интервала весов **b = -1:0.1:1;** % установка интервала пороговых значений

Вычислить и изобразить ошибку нейрона можно следующим образом:

ES = errsurf(p, T, w, b, 'purelin'); % формирование поверхности

ошибки в координатных осях – весов и пороговых значений **plotes(w, b, ES);** % изображение поверхности ошибки

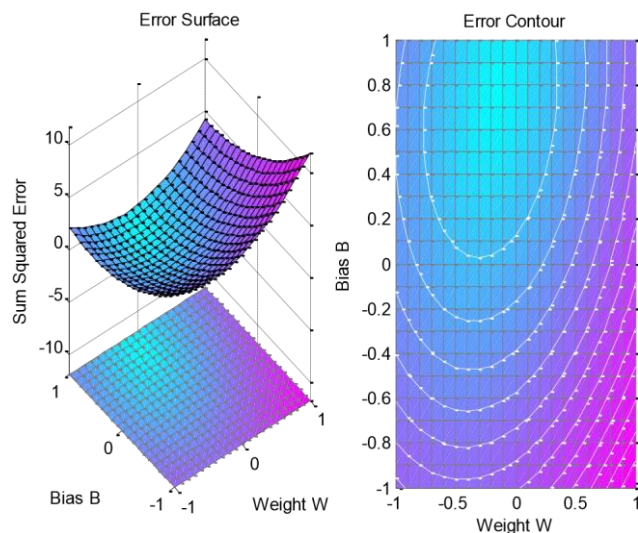


Рисунок 2.11. Изображение поверхности ошибки в координатных осях – весов и пороговых значений

Лучший вес и значение порога – это самые низкие точки на поверхности ошибки. Переопределим уровень обученности и создадим нейрон.

maxlr = 0.20*maxlinlr(p, 'bias'); % возвращение максимального
уровня обученности для линейного слоя с bias, для обучения на основе **p**
net = newlin([-2 2],1,[0], maxlr);

Функция *maxlinlr* находит значение, соответствующее самому быстрому уровню обученности линейной нейронной сети. Возьмем 20% от вычисленного уровня.

Функция *newlin* содержит четыре параметра:

- матрицу интервалов входов размерностью $R \times 2$;
- количество элементов в выходном векторе;

- вектор задержек входов; уровень обучения. □

Установим целевое значение ошибки:

net.trainParam.goal = .001; % установка целевого значения ошибки Получим поверхность ошибки.

ES = errsurf(p, T, w, b, 'purelin'); % формирование поверхности ошибки

plotes(w, b, ES); % изображение поверхности ошибки

subplot (1, 2, 2); % создание 2 координатных осей в графическом окне и установка первых осей активными

Произвольно изменим веса нейрона в установленных интервалах:

net.IW{1, 1} = 0;

net.b{1} = 0.1; % произвольное изменение весов нейрона

[net, tr] = train(net, p, T); % эпоха обучения нейрона, формирование истории обучения tr

Функция *train* выводит на экран историю обучения обученной сети (tr).

График показывает падение ошибки в ходе обучения:

plotperf(tr, net.trainParam.goal); % изображение истории обучения на поверхности ошибки

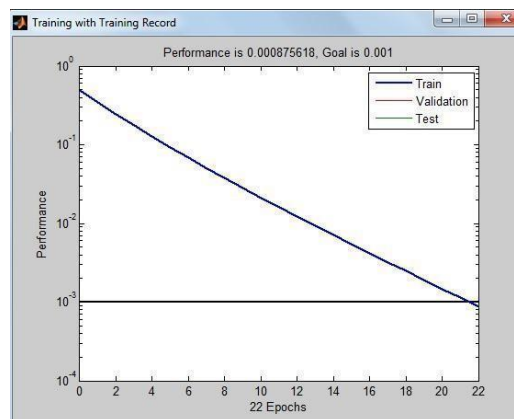


Рисунок 2.12. Изменение среднеквадратической ошибки сети в процессе обучения

p = -1.1;

a = sim(net, p); % формирование выхода обученного нейрона для входа

p Результат очень близок к 1.

Программа работы и методические указания

Изучите возможности нейрона выполнять логические функции двух переменных: И-НЕ (NAND), ИЛИ (OR), ИЛИ-НЕ (NOR), равнозначность (XOR), неравнозначность (NXOR), представленные в таблице 2.1.

Таблица 2.1. Логические функции двух переменных

Входы		Логические функции					
x2	x1	AND	NAND	OR	NOR	XOR	NXOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

Содержание отчета

- цель работы;
- краткое описание действий по пунктам;
- графики по всем пунктам программы;
- выводы по работе.

ПРАКТИЧЕСКАЯ РАБОТА 5. Изучение многослойного нелинейного персептрона и алгоритма обратного распространения ошибки

Цель работы: изучить возможности многослойного персептрона как универсального аппроксиматора и классификатора.

1. Краткие теоретические сведения

В лабораторной работе рассматривается нейронная сеть с прямой передачей сигнала (с прямой связью), то есть сеть, в которой сигналы передаются только в направлении от входного слоя к выходному, и элементы одного слоя связаны со всеми элементами следующего слоя. Важнейшим для реализации нейронных сетей является определение алгоритма обучения сети [57].

В настоящее время одним из самых эффективных и обоснованных методов облужения нейронных сетей является *алгоритм обратного распространения ошибки*, который применим к *однонаправленным многослойным сетям*. В многослойных нейронных сетях имеется множество скрытых нейронов, входы и выходы которых не являются входами и выходами нейронной сети, а соединяют нейроны внутри сети, то есть *скрытые нейроны*. На рисунке 3.1 показана архитектура нейронной сети с прямой передачей сигнала.

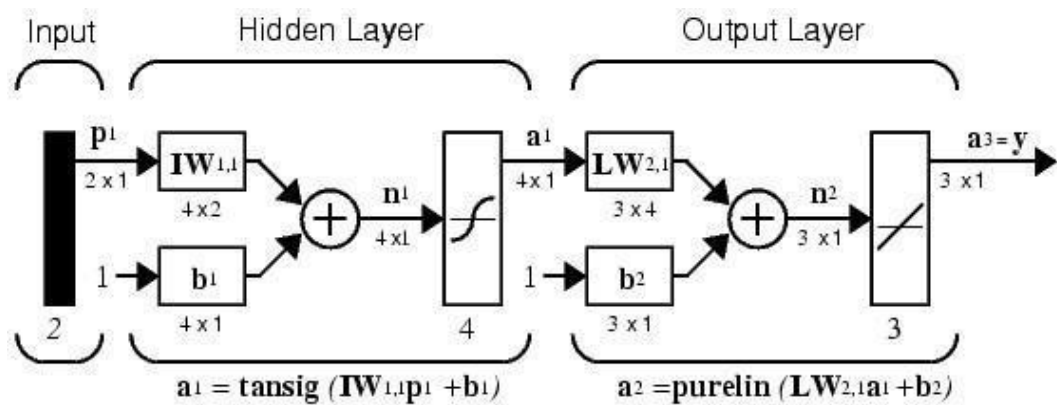


Рисунок 3.1. Схема архитектуры нейронной сети с прямой передачей сигнала

Здесь приняты обозначения: p^1 – вектор входа, $IW^{i,j}$, $LW^{i,j}$ – матрицы весов входа и выхода, b^i – смещение, a^i – выход слоя, y – выход сети, tansig (гиперболическая тангенциальная), purelin (линейная) – соответствующие функции активации.

Веса и смещения определяются с помощью алгоритма обратного распространения ошибок.

Обучение сети обратного распространения требует выполнения следующих операций:

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Скорректировать веса сети так, чтобы минимизировать ошибку.
5. Повторять шаги с 1 по 4 для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

Функция *newff* создает нейронную сеть прямого распространения сигнала, обучаемую с помощью алгоритма обратного распространения ошибки:

$\text{net} = \text{newff}(\text{PR}, [\text{S1 S2 SN1}], \{\text{TF1 TF2 TFN1}\}, \text{BTF}, \text{BLF}, \text{PF})$.

Рассмотрим параметры функции *newff*: PR – матрица интервалов значений для R входных элементов, задаваемых минимальным и максимальным значениями; S_i – размер i -го слоя, для N слоев; TF_i – функция активации i -го слоя, по умолчанию используется функция *tansig* – гиперболический тангенс; BTF – функция обучения сети методом обратного распространения ошибки, по умолчанию используется функция *traingdx*; BLF – функция изменения весов при обучении, по умолчанию используется *learnngdm*; PF – функция измерения ошибки, по умолчанию *mse*. Функция *newff* возвращает многослойную нейронную сеть прямого и обратного распространения сигнала и ошибки соответственно.

Функции активации могут быть выбраны из следующего перечня: гиперболический тангенс *tansig*; логистическая сигмоида *logsig* или линейная функция *purelin*.

2. Создание и обучение нейронной сети с помощью алгоритма обратного распространения ошибки

Создать нейронную сеть, чтобы обеспечить следующее отображение последовательности входа P в последовательность целей T:

$P = [0.10 \ 0.31 \ 0.51 \ 0.72 \ 0.93 \ 1.14 \ 1.34 \ 1.55 \ 1.76 \ 1.96 \ 2.17 \ 2.38 \ 2.59 \ 2.79 \ 3.00];$

$T = [0.1010 \ 0.3365 \ 0.6551 \ 1.1159 \ 1.7632 \ 2.5847 \ 3.4686 \ 4.2115 \ 4.6152 \ 4.6095 \ 4.2887 \ 3.8349 \ 3.4160 \ 3.1388 \ 3.0603];$

```
net = newff([0 3],[4 1],{'tansig' 'purelin'});
```

```
gensim(net)
```

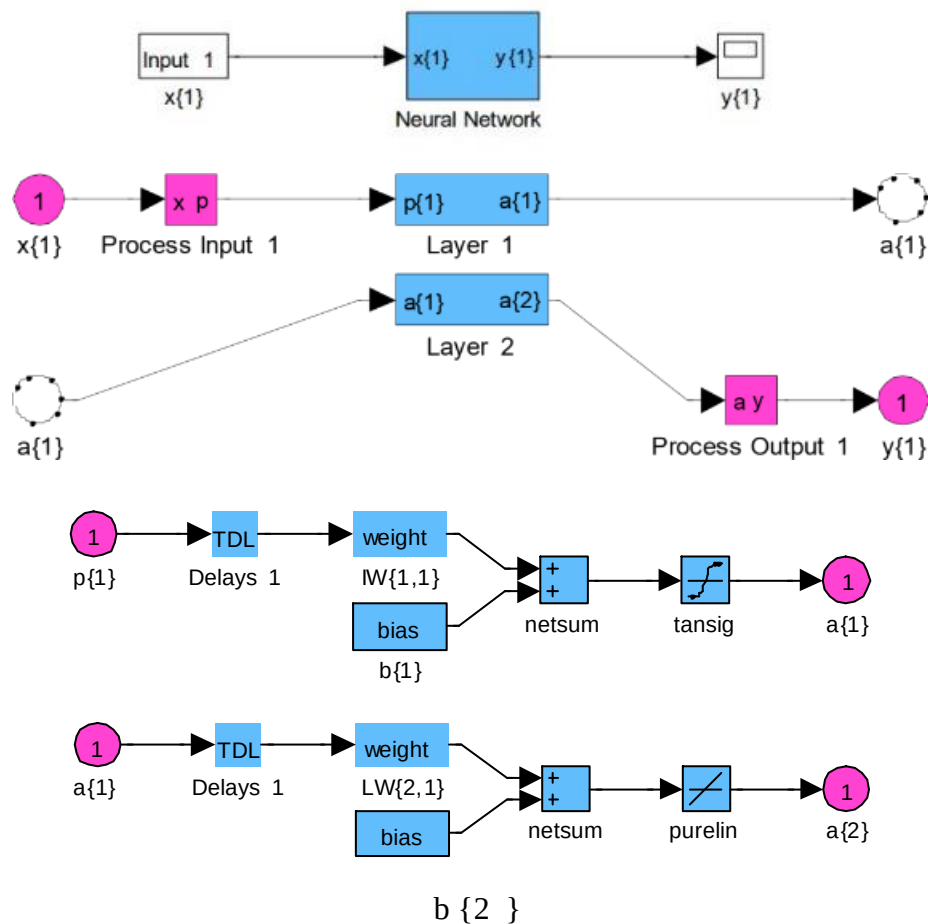


Рисунок 3.2. Структурная схема нейронной сети

Выполним моделирование сети и построим графики сигналов выхода и цели:

```
Y = sim(net,P);
```

```
figure(1), clf plot(P, T, P, Y, 'o'), grid on
```

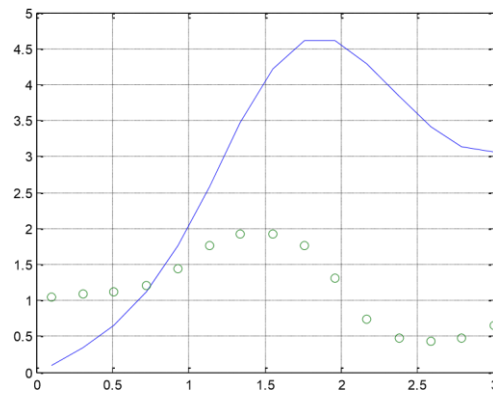


Рисунок 3.3. Графики сигналов выхода и цели

Обучим сеть в течение 50 циклов:

net.trainParam.epochs = 50;

net = train(net, P, T);

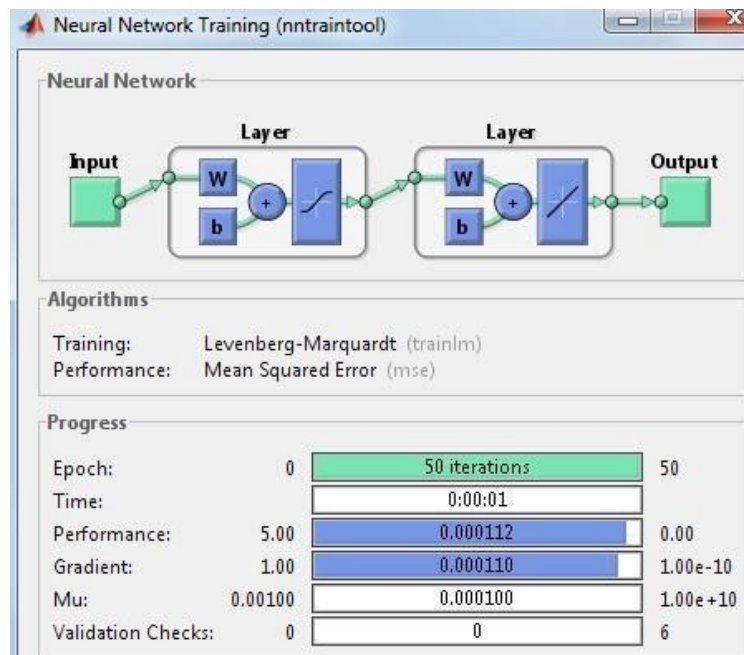


Рисунок 3.4. Окно обучения нейронной сети

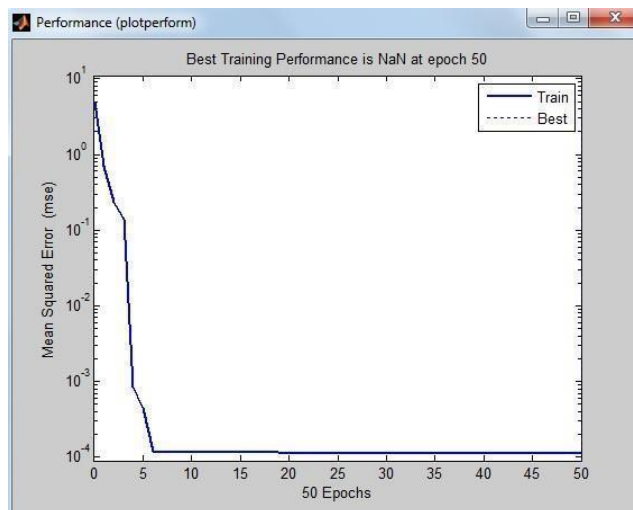


Рисунок 3.5. Изменение ошибки сети в процессе обучения

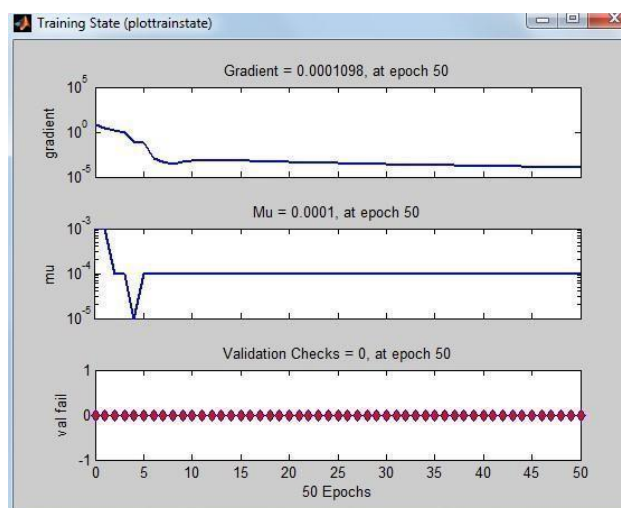


Рисунок 3.6 – Окно состояния обучения

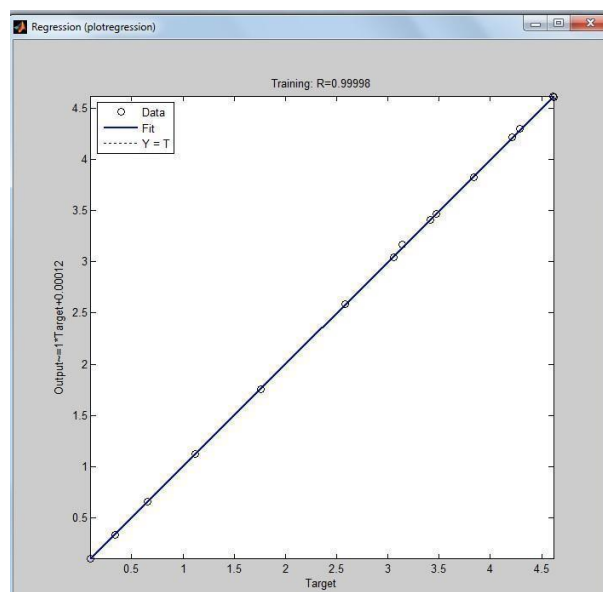


Рисунок 3.7 – Окно линейной регрессии между выходом НС и целями

Выполним моделирование сформированной двухслойной сети, используя обучающую последовательность входа

```
Y = sim(net, P);
```

```
plot(P, T, P, Y, 'o'), grid on
```

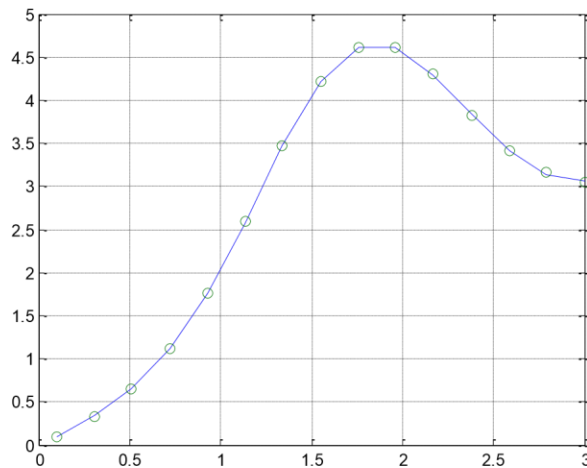


Рисунок 3.8. Результат аппроксимации векторов двухслойным персептроном

Для исследования работы алгоритма обратного распространения ошибки воспользуемся примером, встроенным в MatLab toolbox, набрав команду *demo*.

В появившемся диалоговом окне необходимо последовательно выбирать пункты меню: *toolbox, neural networks, Other Neural Network Design textbook demos, Table of Contents, 10–13, Backpropagation Calculation*.

В примере используется двухслойный персептрон с двумя нелинейными нейронами в первом слое и одним во втором. Действие алгоритма обратного распространения ошибки разбито на следующие шаги: назначение входа и желаемого выхода, прямой проход входного сигнала до выхода, обратное распространение ошибки, изменение весов. Переменные, позволяющие проследить работу алгоритма обратного распространения ошибки, обозначены следующим образом:

P – входной сигнал;

$W_1(i)$ – вектор весов первого слоя, $W_1(1)$ – вес связи, передающий входной сигнал на первый нейрон, а $W_1(2)$ – на второй;

$W_2(i)$ – вектор весов второго слоя, $W_2(1)$ – вес связи, передающий входной сигнал с первого нейрона во второй слой, а $W_2(2)$ – со второго;

$B_2(i)$ – вектор пороговых значений (bias) нейронов первого слоя, $i = 1, 2$;

B_2 – пороговое значение (bias) нейрона второго слоя;

$N_1(i)$ – вектор выходов первого слоя, $i = 1, 2$;

N_2 – выход второго слоя;

$A_1(i)$ – вектор выходных сигналов первого слоя после выполнения функции активации (сигмоиды), $i = 1, 2$;

A_2 – выход второго слоя после выполнения функции активации (линейной);

Lr – коэффициент обучаемости.

Пусть входной сигнал $P = 1.0$, а желаемый выход

$$t = 1 + \sin\left(p * \frac{\pi}{4}\right) = 1.707$$

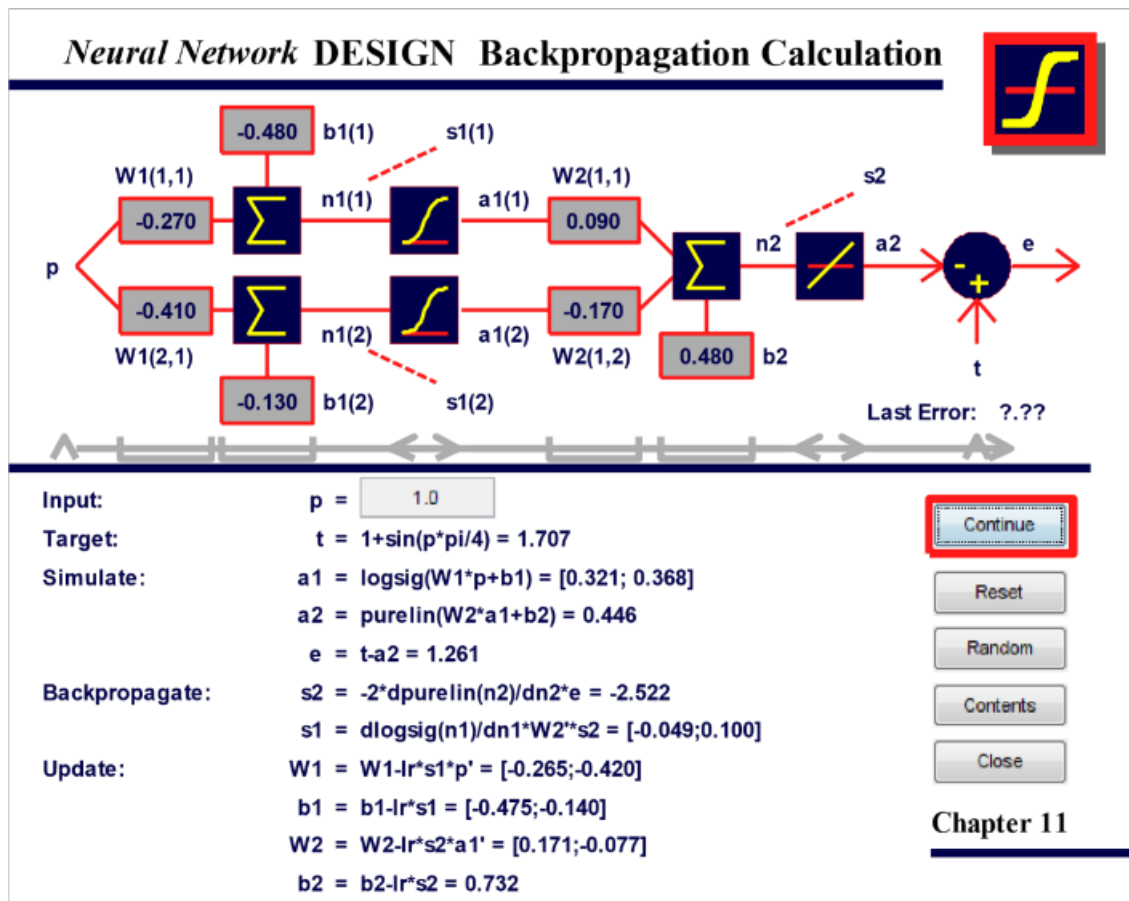


Рисунок 3.9. Демо пример работы алгоритма *Backpropagation*

3. Исследование градиентных алгоритмов обучения нейронных сетей

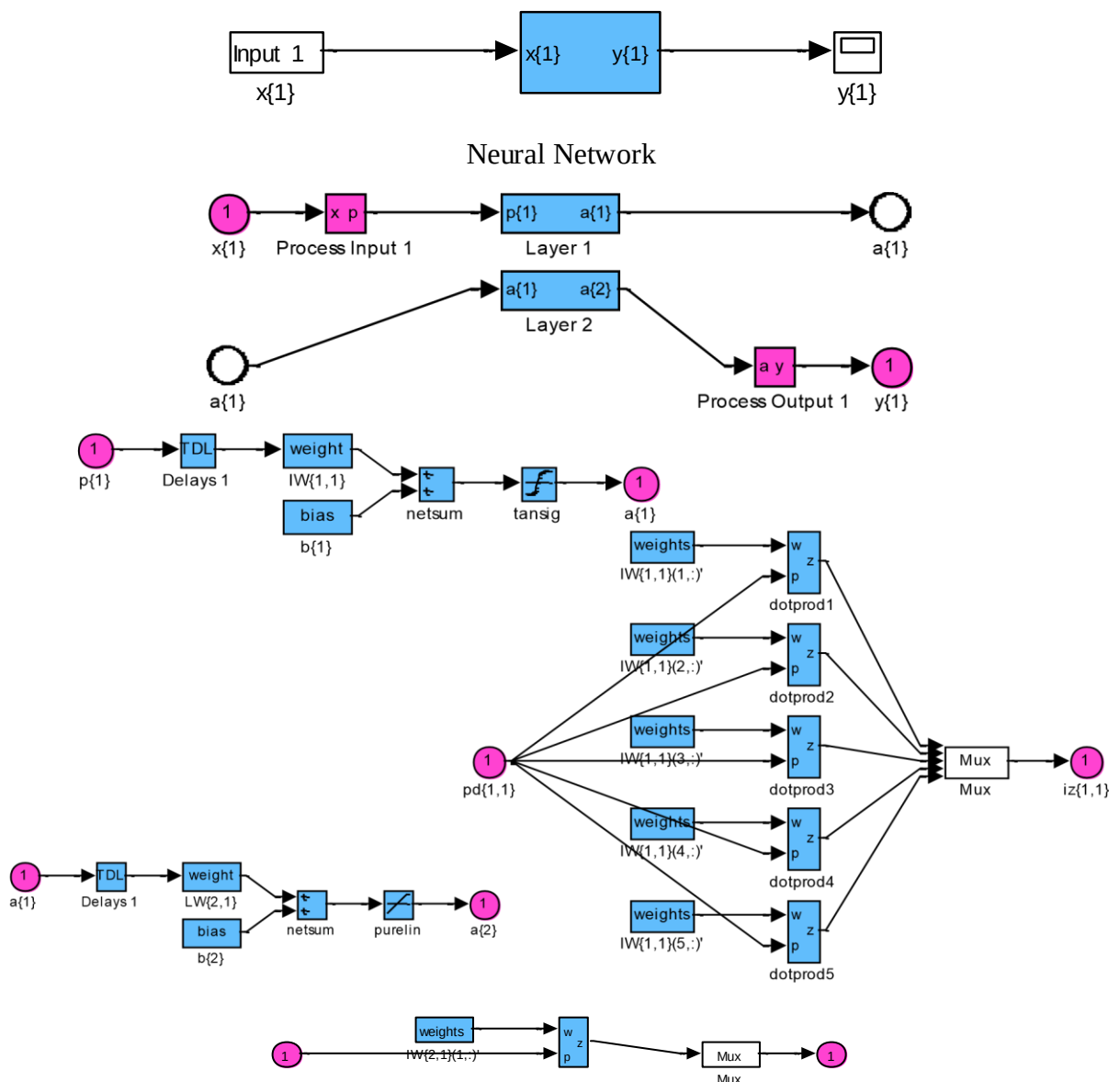
Алгоритмы обучения, как правило, функционируют пошагово; и эти шаги принято называть *эпохами* или *циклами*. На каждом цикле на вход сети последовательно подаются все элементы обучающей последовательности, затем вычисляются выходные значения сети, сравниваются с целевыми и вычисляется функционал ошибки. Значения функционала, а также его градиента используются для корректировки весов и смещений, после чего все действия повторяются. Начальные значения весов и смещений выбираются случайным образом, а процесс обучения прекращается, когда выполнено определенное количество циклов либо когда ошибка достигнет некоторого малого значения или перестанет уменьшаться [54].

При такой формализации задачи обучения предполагаются известными желаемые (целевые) реакции сети на входные сигналы, что ассоциируется с присутствием учителя, а поэтому такой процесс обучения называют *обучением с учителем*. Для некоторых типов нейронных сетей задание целевого сигнала не требуется, и в этом случае процесс обучения называют *обучением без учителя*.

Алгоритм GD. Алгоритм GD, или алгоритм градиентного спуска, используется для такой корректировки весов и смещений, чтобы минимизировать функционал ошибки, т. е. обеспечить движение по поверхности функционала в направлении, противоположном градиенту функционала по настраиваемым параметрам.

net = newff([-1 1;-1 1],[5,1],{'tansig','purelin'},'traingd'); % создание двухслойной НС с прямой передачи сигнала с сигмоидальным и линейным слоями для обучения ее на основе метода обратного распространения ошибки **gensim(net)** %

Структурная схема модели нейронной сети показана на рис 3.10.



ad{2,1} dotprod1 lz{2,1}

Рисунок 3.10. Структурная схема модели нейронной сети

net.biases{1,1}.learnFcn='learngd'; % задание имени функции настройки learnFcn, соответствующее алгоритму градиентного спуска, в данном случае это Мфункция **learngd**

net.biases{2,1}.learnFcn='learngd';

net.layerWeights{2,1}.learnFcn='learngd';

net.inputWeights{1,1}.learnFcn='learngd';

net.layerWeights{2,1}.learnParam.lr = 0.2; % установка параметра скорости настройки

x=-1:0.1:1; % задание множества входов

y=x;

p=[x;y];

p=num2cell(p,1); % поскольку используется последовательный способ обучения, преобразуем массив входов в массив ячеек

t=sin(x.^2)./cos(y.^2); % задание множества целей

t=num2cell(t,1); % преобразование массива целей в массивы ячеек

net.adaptParam.passes=300; % число проходов при последовательном обучении

tic, [net, a, e]=adapt(net, p, t); toc % настройка параметров

НС, используя процедуру адаптации Elapsed time is 18.555497 seconds.

a=sim(net, p) % моделирование НС для проверки качества обучения **a =**

[1.2764] [1.1352] [0.8998] [0.5909] [0.2899] [0.0750] [-0.0316] [-0.0387]

[0.0315] [0.1196] [0.1581] [0.1359] [0.0848] [0.0430] [0.0431] [0.1093] [0.2564]

[0.4907] [0.8103] [1.2054] [1.6578]

mse(e) % среднеквадратическая ошибка **ans = 0.0334**

[x,y]=meshgrid(x,y);

z=sin(x.^2)./cos(y.^2);

surf(x,y,z) % построение поверхности целевой функции

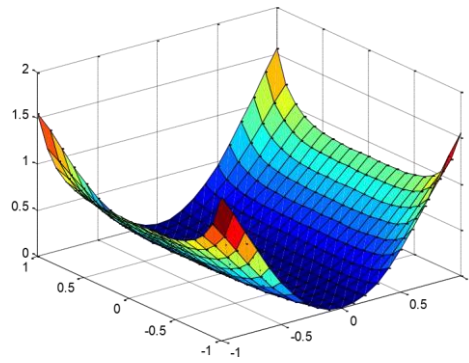


Рисунок 3.11. График поверхности целевой функции

```
a=cat(1,a{:}); % преобразование массива ячеек результата
```

моделирования НС в массив чисел

```
[a]=meshgrid(a);
```

```
surf(x,y,a) % построение поверхности функции аппроксимируемой нейронной сетью
```

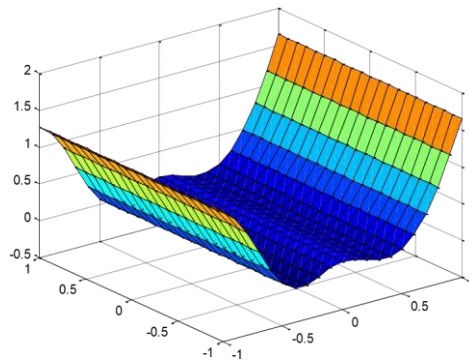


Рисунок 3.12. График поверхности функции аппроксимируемой нейронной сетью

Групповое обучение. Для обучения сети на основе алгоритма GD необходимо использовать М-функцию `traingd` взамен функции настройки `learngd`. В этом случае нет необходимости задавать индивидуальные функции обучения для весов и смещений, а достаточно указать одну обучающую функцию для всей сети.

```
net = newff([-1 1;-1 1],[5,1],{'tansig','purelin'},'traingd');
```

% вновь создадим ту же двухслойную нейронную сеть прямой передачи сигнала с сигмоидальным и линейным слоями для обучения по методу обратного распространения ошибки

```
net.trainParam.show = 50;
```

% show – интервал вывода информации

```
net.trainParam.lr = 0.05;
```

% lr – параметр скорости настройки

```
net.trainParam.goal = 1e-005;
```

% goal – предельное значение критерия обучения

x=-1:0.1:1;

y=x;

p=[x;y];

t=sin(x.^2)./cos(y.^2); tic,

net=train(net,p,t); toc Elapsed time is 4.796924 seconds.

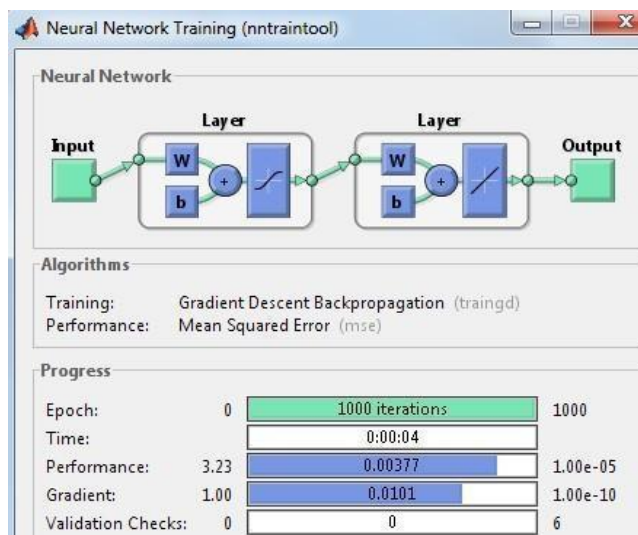


Рисунок 3.13. Окно обучения нейронной сети

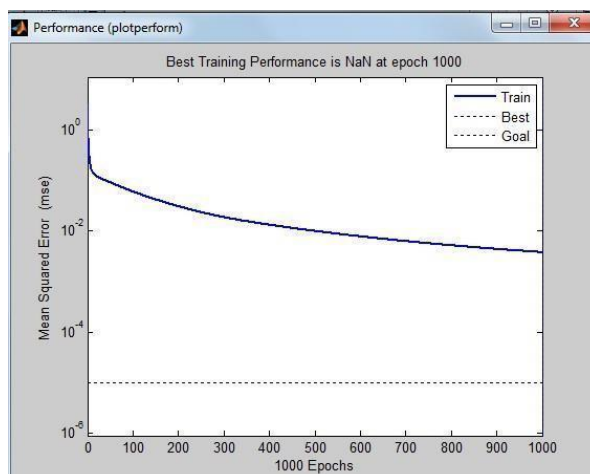


Рисунок 3.14. График изменения ошибки в зависимости от числа выполненных циклов обучения

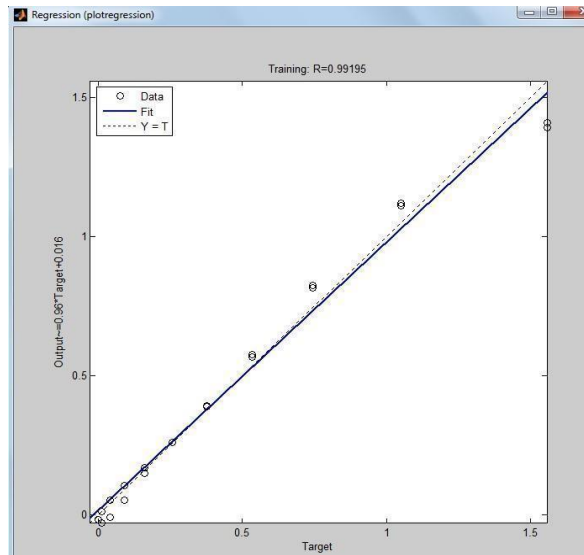


Рисунок 3.15. Окно линейной регрессии между выходом НС и целями $a=\text{sim}(\text{net}, p)$

$a =$

1.4087 1.1199 0.8169 0.5667 0.3896 0.2606 0.1488 0.0524 -0.0088 -0.0281 -0.0165 0.0132
0.0542 0.1057 0.1718 0.2621 0.3915 0.5765 0.8232 1.1106 1.3917

$e=t-a$

0.1487 -0.0695 -0.0723 -0.0333 -0.0132 -0.0052 0.0126 0.0378 0.0488 0.0381 0.0165 -0.0032 -
0.0142 -0.0155 -0.0105 -0.0068 -0.0151 -0.0431 -0.0786 -0.0601 0.1657

$\text{mse}(e)$

ans = 0.0038

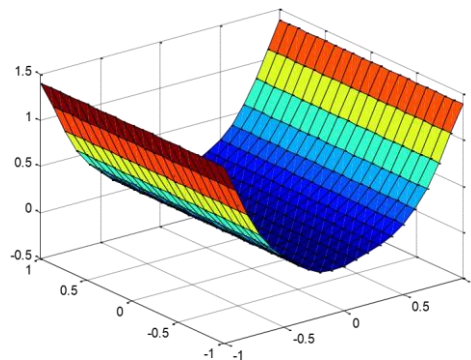


Рисунок 3.16. График поверхности функции аппроксимируемой нейронной сетью

Более тщательно ознакомиться с методом градиентного спуска можно с помощью демонстрационной программы `nn12sd1`, которая иллюстрирует работу алгоритма GD.

Алгоритм GDM. Алгоритм GDM, или алгоритм градиентного спуска с возмущением, предназначен для настройки и обучения сетей прямой передачи. Этот алгоритм позволяет преодолевать локальные неровности поверхности ошибки и не останавливаться в локальных минимумах. С учетом возмущения метод обратного распространения ошибки реализует следующее соотношение для приращения вектора настраиваемых параметров:

$$\Delta w_k = mc\Delta w_{k-1} + (1 - mc)lrq_k, \quad (3.1)$$

где Δw_k – приращение вектора весов; mc – параметр возмущения; lr – параметр скорости обучения; g_k – вектор градиента функционала ошибки на k -й итерации.

Если параметр возмущения равен 0, то изменение вектора настраиваемых параметров определяется только градиентом, если параметр равен 1, то текущее приращение равно предшествующему как по величине, так и по направлению.

```
net = newff([-1 1;-1 1],[5,1],{'tansig','purelin'},'traingdm');
```

```
net.biases{1,1}.learnFcn='learngdm';
```

```
% задание имени функции
```

```
настройки learnFcn, соответствующее алгоритму градиентного спуска с  
возмущением – М-функция learngdm
```

```
net.biases{2,1}.learnFcn='learngdm';
```

```
net.layerWeights{2,1}.learnFcn='learngdm';
```

```
net.inputWeights{1,1}.learnFcn='learngdm';
```

```
net.layerWeights{2,1}.learnParam.lr=0.2;
```

```
% увеличим значение
```

```
параметра скорости обучения до 0.2
```

```
x=-1:0.1:1;
```

```
y=x;
```

```
p=[x;y];
```

```
p=num2cell(p,1);
```

```
t=sin(x.^2)./cos(y.^2);
```

```
t=num2cell(t,1);
```

```
net.adaptParam.passes=300;
```

```
% значение проходов tic,
```

```
[net,a,e]=adapt(net,p,t); toc
```

```
Elapsed time is 19.187137 seconds.
```

a =

```
[0.8683] [1.0634] [1.2363] [1.2927] [1.1903] [0.9339] [0.5665] [0.1546] [-0.2283] [-0.5175] [-  
0.6697] [-0.6692] [-0.5286] [-0.2830] [0.0194] [0.3291] [0.6065] [0.8297] [0.9997] [1.1413]  
[1.3017] mse(e)
```

ans = 0.2196

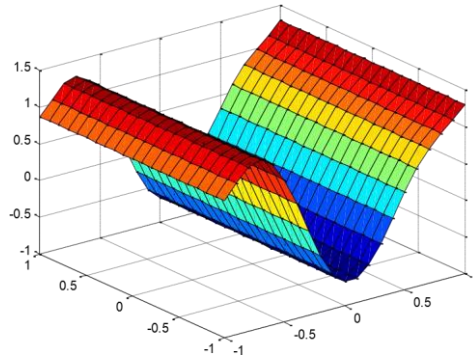


Рисунок 3.17. График поверхности функции аппроксимируемой нейронной сетью

Результаты аппроксимации оказались хуже результатов работы НС с алгоритмом обучения GD и намного дольше.

Групповое обучение. Альтернативой последовательной адаптации является групповое обучение, которое основано на применении функции `train`. В этом режиме параметры сети модифицируются только после того, как реализовано все обучающее множество, и градиенты, рассчитанные для каждого элемента множества, суммируются, чтобы определить приращения настраиваемых параметров.

Для обучения сети на основе алгоритма GDM необходимо использовать Мфункцию `traingdm` взамен функции настройки `learnngdm`. Различие этих двух функций состоит в следующем. Алгоритм функции `traingdm` суммирует градиенты, рассчитанные на каждом цикле обучения, а параметры модифицируются только после того, как все обучающие данные будут представлены. Если проведено N циклов обучения и для функционала ошибки оказалось выполненным условие $J_N \geq 1.04J_{N-1}$, то параметр возмущения `mc` следует установить в 0.

```
net = newff([-1 1;1],[5,1],{'tansig','purelin'},'traingdm');
```

```
net.trainParam.epochs=1000;
```

```
net.trainParam.goal=1e-5;
```

```
net.trainParam.lr=0.05;
```

```
net.trainParam.mc =0.9;
```

```
% по сравнению с функцией traingd здесь
```

```
добавляется только 1 параметр возмущения mc
```

```
net.trainParam.show=50;
```

```
x=-1:0.1:1;
```

```
y=x;
```

```
p=[x;y];
```

```
t=sin(x.^2)./cos(y.^2); tic,
```

```
net=train(net,p,t); toc
```

Elapsed time is 5.533868 seconds.

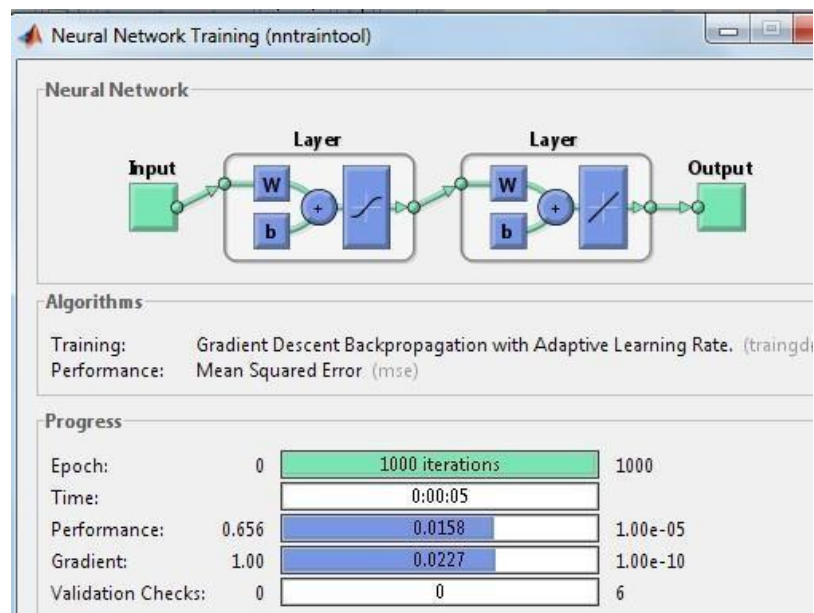


Рисунок 3.18. Окно обучения нейронной сети

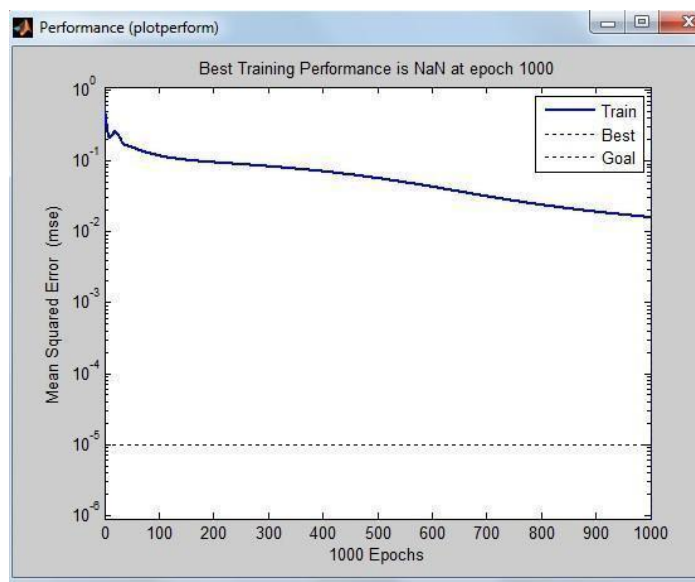


Рисунок 3.19. График изменения ошибки обучения в зависимости от числа выполненных циклов обучения

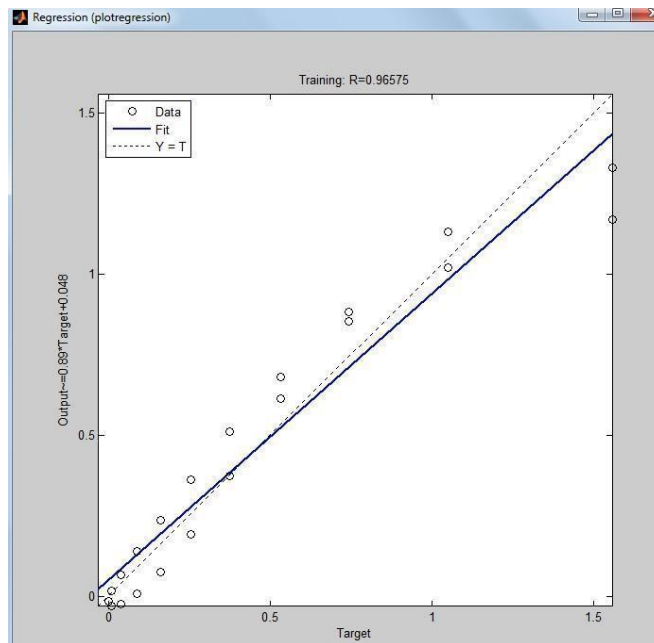


Рисунок 3.20. Окно линейной регрессии между выходом НС и целями $a=\text{sim}(\text{net},p)$; $e=t-a$; $\text{mse}(e)$ ans = 0.0158

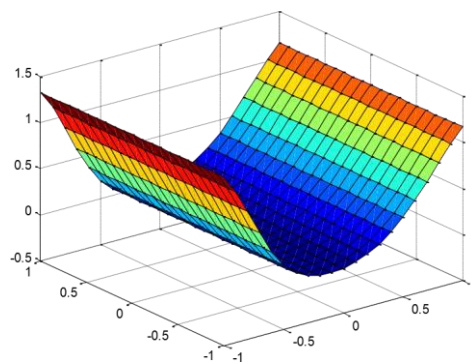


Рисунок 3.21. График поверхности функции аппроксимируемой нейронной сетью

Более тщательно ознакомиться с методом градиентного спуска с возмущением можно с помощью демонстрационной программы `nnd12mo`, которая иллюстрирует работу алгоритма GDM.

Практика применения описанных выше алгоритмов градиентного спуска показывает, что эти алгоритмы слишком медленны для решения реальных задач. Ниже обсуждаются алгоритмы группового обучения, которые сходятся в десятки и сотни раз быстрее. Ниже представлены 2 разновидности таких алгоритмов: один основан на стратегии выбора параметра скорости настройки и реализован в виде алгоритма GDA, другой – на стратегии выбора шага с помощью порогового алгоритма обратного распространения ошибки и реализован в виде алгоритма Rprop.

Алгоритм GDA. Алгоритм GDA, или алгоритм градиентного спуска с выбором параметра скорости настройки, использует эвристическую стратегию изменения этого параметра в процессе обучения.

Эта стратегия заключается в следующем. Вычисляются выход и погрешность инициализированной нейронной сети. Затем на каждом цикле обучения вычисляются

новые значения настраиваемых параметров и новые значения выходов и погрешностей. Если отношение нового значения погрешности к прежнему превышает величину `max_perf_inc` (по умолчанию 1.04), то новые значения настраиваемых параметров во внимание не принимаются. При этом параметр скорости настройки уменьшается с коэффициентом `lr_dec` (по умолчанию 0.7). Если новая погрешность меньше прежней, то параметр скорости настройки увеличивается с коэффициентом `lr_inc` (по умолчанию 1.05).

Эта стратегия способствует увеличению скорости и сокращению длительности обучения.

Функция `traingda` характеризуется следующими параметрами, заданными по умолчанию:

`net.trainParam`

`show`: 50

`showWindow`: 1

`showCommandLine`: 0

`epochs`: 300

`time`: Inf

`goal`: 1.0000e-005

`max_fail`: 6

`lr`: 0.0500

`lr_inc`: 1.0500

`lr_dec`: 0.7000

`max_perf_inc`: 1.0400

`min_grad`: 1.0000e-010

`mc`: 0.9000

Алгоритм GDA в сочетании с алгоритмом GD определяет функцию обучения `traingda`, а в сочетании с алгоритмом GDM - функцию обучения `traingdx`.

Вновь обратимся к той же нейронной сети, но будем использовать функцию обучения `traingda`:

`net = newff([-1 1;-1 1],[5,1],{'tansig','purelin'},'traingda');`

`net.trainParam.epochs=300;`

`net.trainParam.goal=1e-5;`

`net.trainParam.lr=0.05;`

`net.trainParam.mc=0.9;`

`net.trainParam.show=50;`

```

x=-1:0.1:1;
y=x;
p=[x;y];
t=sin(x.^2)./cos(y.^2); tic,
net=train(net,p,t);toc
Elapsed time is 1.706964 seconds.

```

На рисунке 3.22 приведен график изменения ошибки обучения в зависимости от числа выполненных циклов.

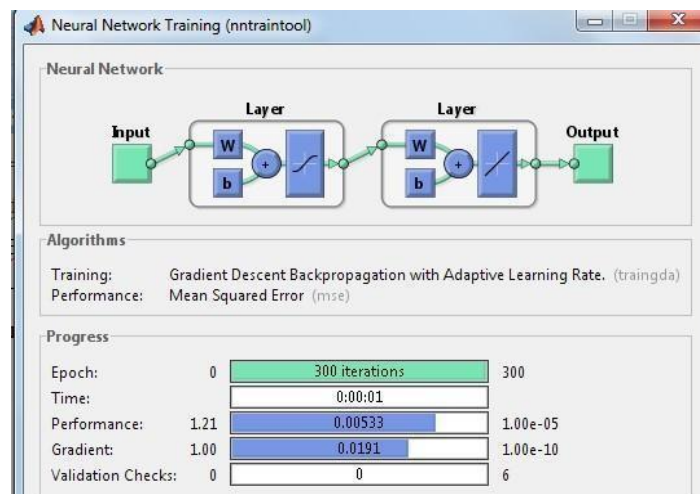


Рисунок 3.22. Окно обучения нейронной сети

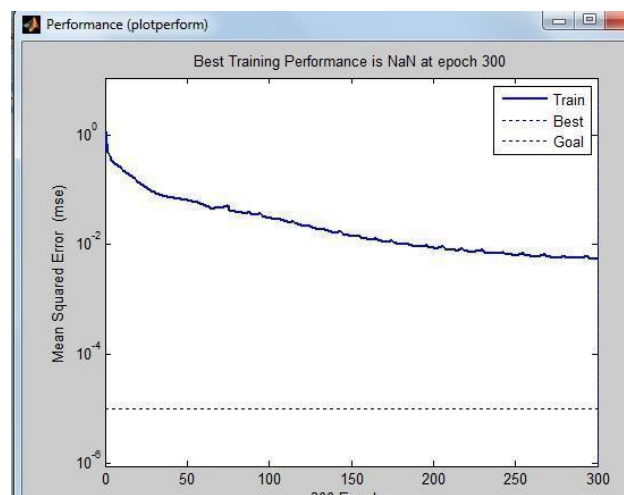


Рисунок 3.23. График изменения ошибки обучения в зависимости от числа выполненных циклов обучения

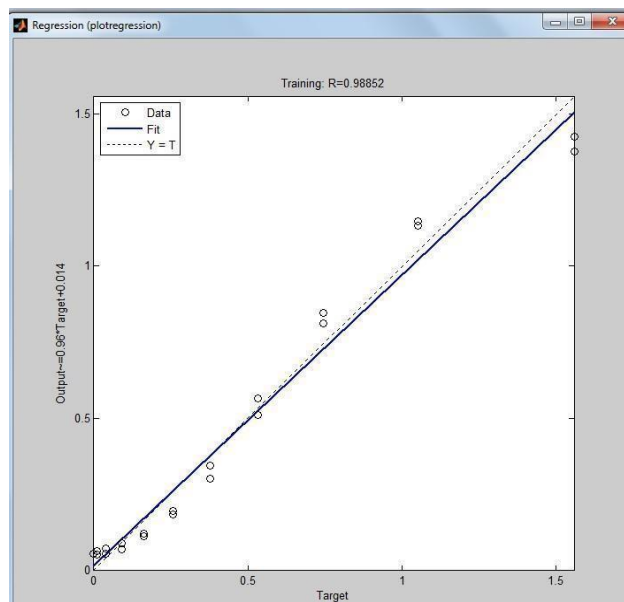


Рисунок 3.24. Окно линейной регрессии между выходом НС и целями

a=sim(net,p);

e=t-a;

mse(e)

0.0053

Нетрудно заметить, что количество циклов обучения по сравнению с предыдущим примером сократилось практически в 3 раза при уменьшении погрешности обучения.

Демонстрационная программа *nn12v1* иллюстрирует производительность алгоритма с переменным параметром скорости настройки.

Алгоритм Rprop. Алгоритм Rprop, или пороговый алгоритм обратного распространения ошибки, реализует следующую эвристическую стратегию изменения шага приращения параметров для многослойных нейронных сетей.

Многослойные сети обычно используют сигмоидальные функции активации в скрытых слоях. Эти функции относятся к классу функций со сжимающим отображением, поскольку они отображают бесконечный диапазон значений аргумента в конечный диапазон значений функции. Сигмоидальные функции характеризуются тем, что их наклон приближается к нулю, когда значения входа нейрона существенно возрастают. Следствием этого является то, что при использовании метода наискорейшего спуска величина градиента становится малой и приводит к малым изменениям настраиваемых параметров, даже если они далеки от оптимальных значений.

Цель порогового алгоритма обратного распространения ошибки Rprop (Resilient propagation) состоит в том, чтобы повысить чувствительность метода при больших значениях входа функции активации. В этом случае вместо значений самих производных используется только их знак.

Значение приращения для каждого настраиваемого параметра увеличивается с коэффициентом *delt_inc* (по умолчанию 1.2) всякий раз, когда производная функционала

ошибки по данному параметру сохраняет знак для двух последовательных итераций. Значение приращения уменьшается с коэффициентом `delt_dec` (по умолчанию 0.5) всякий раз, когда производная функционала ошибки по данному параметру изменяет знак по сравнению с предыдущей итерацией. Если производная равна 0, то приращение остаётся неизменным. Поскольку по умолчанию коэффициент увеличения приращения составляет 20%, а коэффициент уменьшения – 50%, то в случае попеременного увеличения и уменьшения общая тенденция будет направлена на уменьшение шага изменения параметра. Если параметр от итерации к итерации изменяется в одном направлении, то шаг изменения будет постоянно возрастать.

Алгоритм Rprop определяет функцию обучения `trainrp`.

Функция `trainrp` характеризуется следующими параметрами, заданными по умолчанию:

`net.trainParam`

`ans = show: 25`

`showWindow: 1`

`showCommandLine: 0`

`epochs: 1000`

`time: Inf`

`goal: 0`

`max_fail: 6`

`min_grad: 1.0000e-010`

`delt_inc: 1.2000`

`delt_dec: 0.5000`

`delta0: 0.0700`

`deltamax: 50`

Здесь `epochs` – максимальное количество циклов обучения; `show` – интервал вывода информации, измеренный в циклах; `goal` – предельное значение критерия обучения; `time` – предельное время обучения; `min_grad` – минимальное значение градиента; `max_fail` – максимально допустимый уровень превышения ошибки контрольного подмножества по сравнению с обучающим; `delt_inc` – коэффициент увеличения шага настройки; `delt_dec` – коэффициент уменьшения шага настройки; `delta0` – начальное значение шага настройки; `deltamax` – максимальное значение шага настройки.

Вновь обратимся к сети, показанной на рисунке 3.10, но будем использовать функцию обучения `trainrp`:

`net=newff([-1 2;0 5],[3,1],{'tansig','purelin'},'trainrp');`

Установим следующие значения этих параметров:

```

net.trainParam.show = 10;

net.trainParam.epochs = 300;

net.trainParam.goal = 1e-5;

x=-1:0.1:1;

y=x;

p=[x;y];

t=sin(x.^2)./cos(y.^2); tic,

net=train(net,p,t); toc

Elapsed time is 1.928695 seconds.

```

На рисунке 3.26 приведен график изменения ошибки обучения в зависимости от числа выполненных циклов обучения.

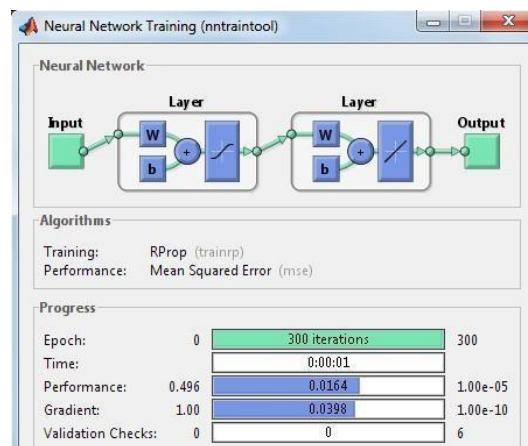


Рисунок 3.26. Окно обучения нейронной сети

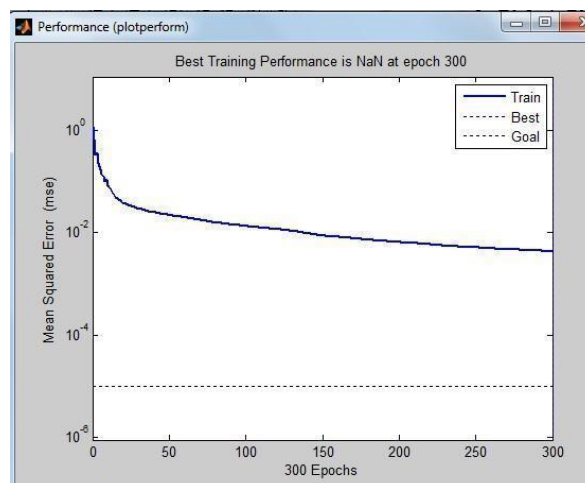


Рисунок 3.26. График изменения ошибки обучения в зависимости от числа выполненных циклов обучения $a=\text{sim}(\text{net},p)$; $e=t-a$; $\text{mse}(e)$ 0.0043

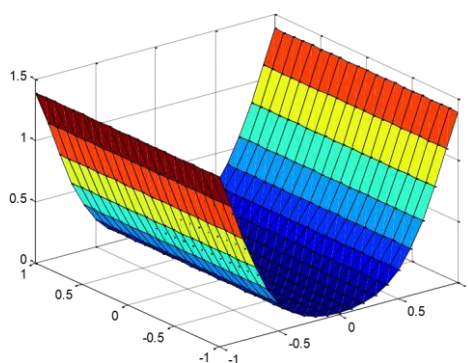


Рисунок 3.27. График поверхности функции аппроксимируемой нейронной сетью

Нетрудно заметить, что количество циклов обучения по сравнению с алгоритмом GDA сократилось практически еще в 3 раза и составило по отношению к алгоритму GD значение, близкое к 9.

Программа работы и методические указания

При заданном преподавателем значении p , рассчитайте в командном окне MatLab прямое распространение входного сигнала, обратное распространение ошибки и изменение весов на первой итерации алгоритма *Backpropagation*.

Сравните с результатами, полученными в *Backpropagation Calculation*.

Контрольные вопросы

1. Каким алгоритмом обучают многослойные нейронные сети?
2. Из каких основных этапов состоит алгоритм обратного распространения ошибки?
3. Почему алгоритм обратного распространения ошибки относится к классу алгоритмов градиентного спуска?
4. Как влияет функция принадлежности на правило изменения весов в обратном алгоритме распространения ошибки?

Содержание отчета

- цель работы;
- краткое описание действий по пунктам;
- графики по всем пунктам программы;
- расчеты и выводы по работе.

ПРАКТИЧЕСКАЯ РАБОТА 6. Применение нейронных сетей для распознавания образов

Цель: изучить интерфейс пакета Neural Pattern Recognition Tool.

Порядок и последовательность выполнения работы

1. Постановка задачи распознавания образов
2. Интерфейс пакета распознавания образов

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ

1. Постановка задачи распознавания образов

Задача распознавания образов состоит в отнесении входного набора данных, представляющего распознаваемый объект, к одному из заранее известных классов. Это может быть задача распознавания рукописных и печатных символов при оптическом вводе в ЭВМ, распознавание типов клеток крови, распознавание речи и пр.

Распознавание образов - это отнесение исходных данных к определенному классу с помощью выделения существенных признаков, характеризующих эти данные, из общей массы несущественных данных.

Постановка задачи распознавания образов. Дано множество объектов. Относительно них необходимо провести классификацию. Множество представлено подмножествами, которые называются классами. Заданы: информация о классах, описание всего множества и описание информации об объекте, принадлежность которого к определенному классу неизвестна. Требуется по имеющейся информации о классах и описании объекта установить - к какому классу относится этот объект.

2. Интерфейс пакета распознавания образов

Распознавание образов является областью наиболее эффективного применения нейронных сетей. Рассмотрим инструментальные средства пакета Neural Pattern Recognition Tool пакета MATLAB, предназначенные для распознавания образов. Так и в случае приближения функций, рассмотрим визуальные средства распознавания образов и возможность генерирования созданных компьютерных моделей нейронных сетей распознавания образов в скрипты.

Для запуска мастера создания нейронной сети распознавания образов можно нажать кнопку окна выбора инструментальных средств нейронных сетей Pattern recognition app (рисунок 1.1) или в окне команд набрать:

```
>>nprtool,
```

после этого будет выведено окно Neural Pattern Recognition, показанное на рисунке 4.1.

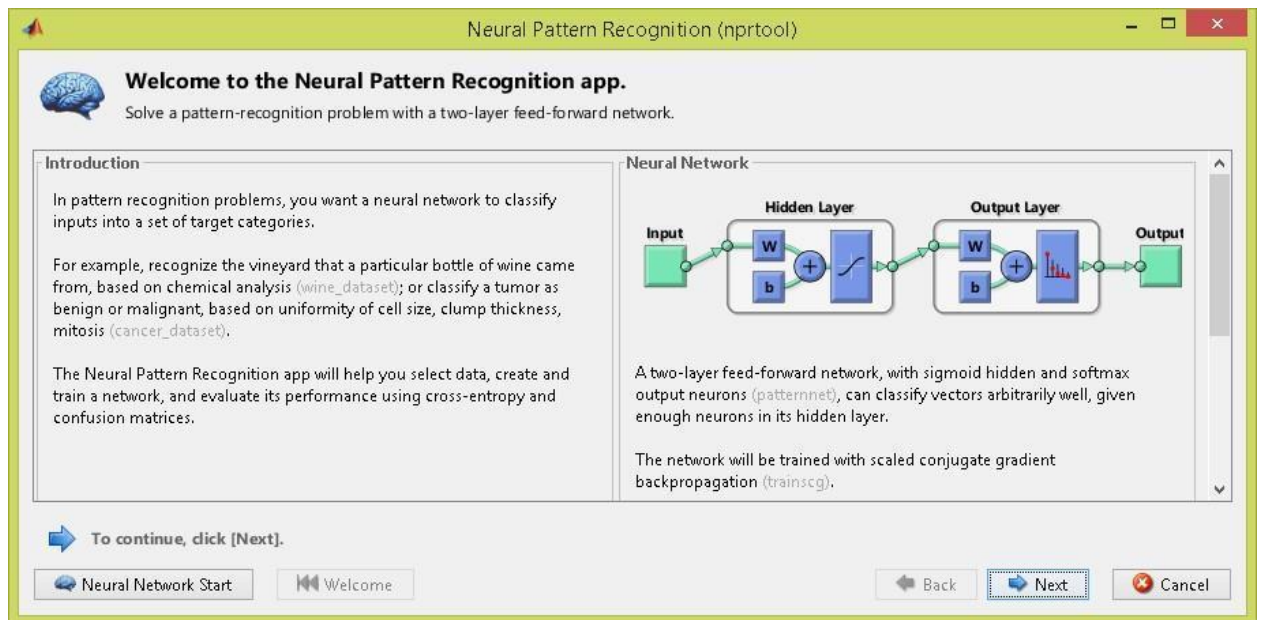


Рисунок 4.1 – Начальное окно мастера Neural Pattern Recognition

Нажатие кнопки Next вызывает следующее окно мастера Select Data, посвященное заданию исходных данных (рисунок 4.2).

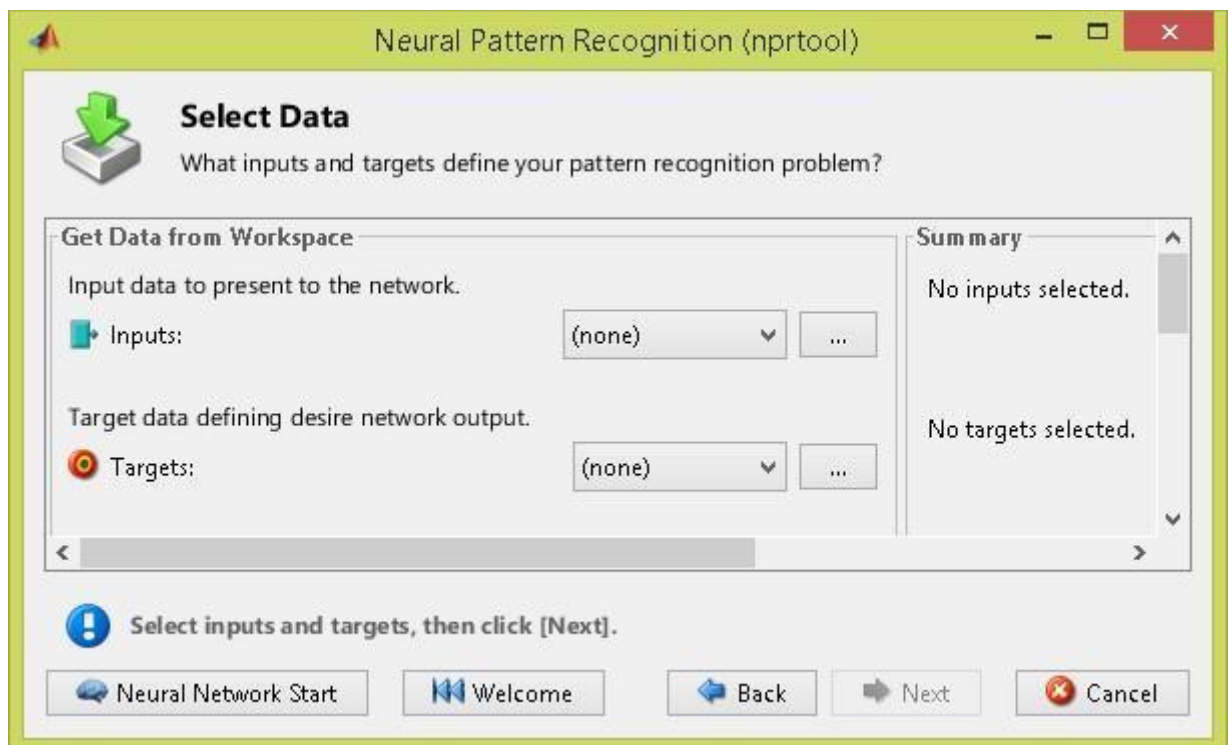


Рисунок 4.2 – Окно задания исходных данных мастера

Работу мастера создания нейронной сети распознавания образов рассмотрим на демонстрационном примере. Для выбора одного из демонстрационных примеров необходимо нажать кнопку Load Example Data

Set окна мастера выбора исходных данных. При этом будет выведено окно Pattern Recognition Data Set Chooser, показанное на рисунке 4.3, в левой половине которого выделен демонстрационный пример простых классов

(Simple Classes).

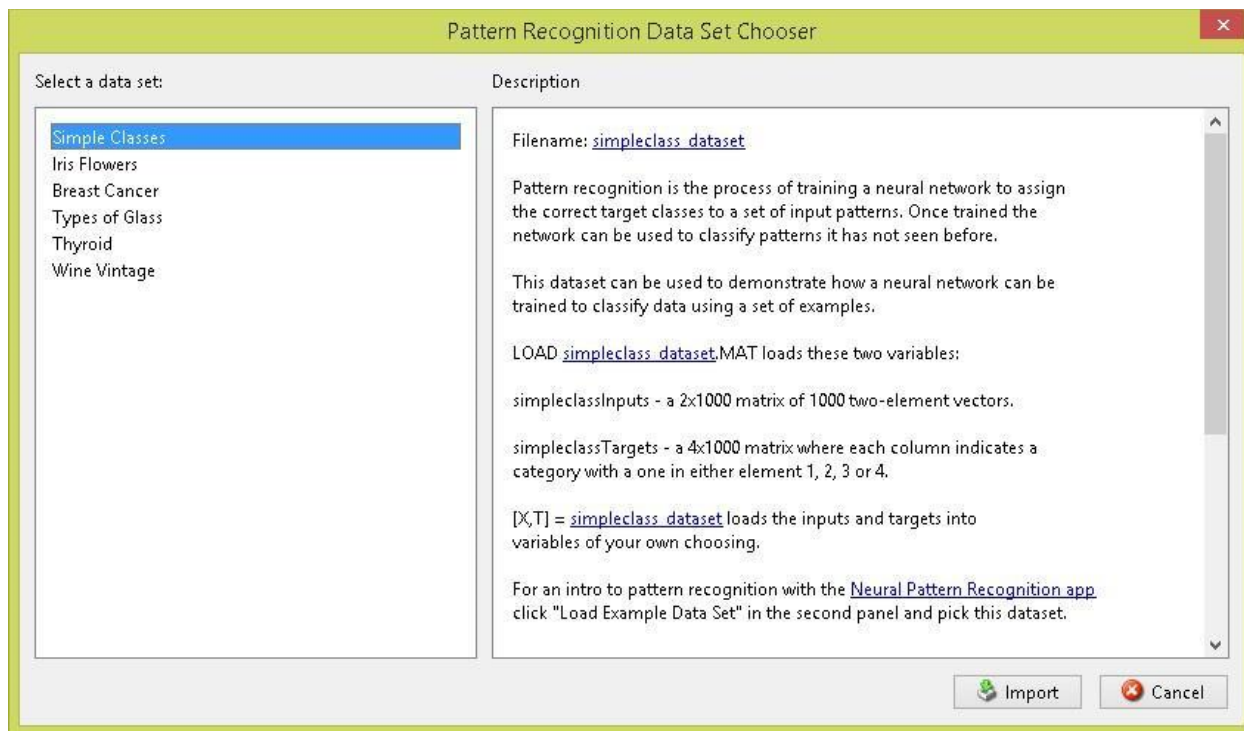


Рисунок 4.3 – Окно выбора данных демонстрационных примеров

Импорт исходных данных, распределение проверочных тестовых и данных обучающей выборки, задание параметров архитектуры нейронной сети обучение сети осуществляется по аналогии с работой мастера создания нейронной сети аппроксимации функции. После нажатия кнопки Plot Confusion выводиться результаты анализа функционирования сети с графиком правильности распознавания для данных обучающей выборки, проверочных, тестовых и обобщенных наборах (рисунок 4.4). По диагонали в зеленых клетках указан процент правильного распознавания для каждого образа, в красных – ошибочных, в синей, нижней правой клетке – общей процент правильного распознавания. Из графиков видно, что сеть обучена правильно.

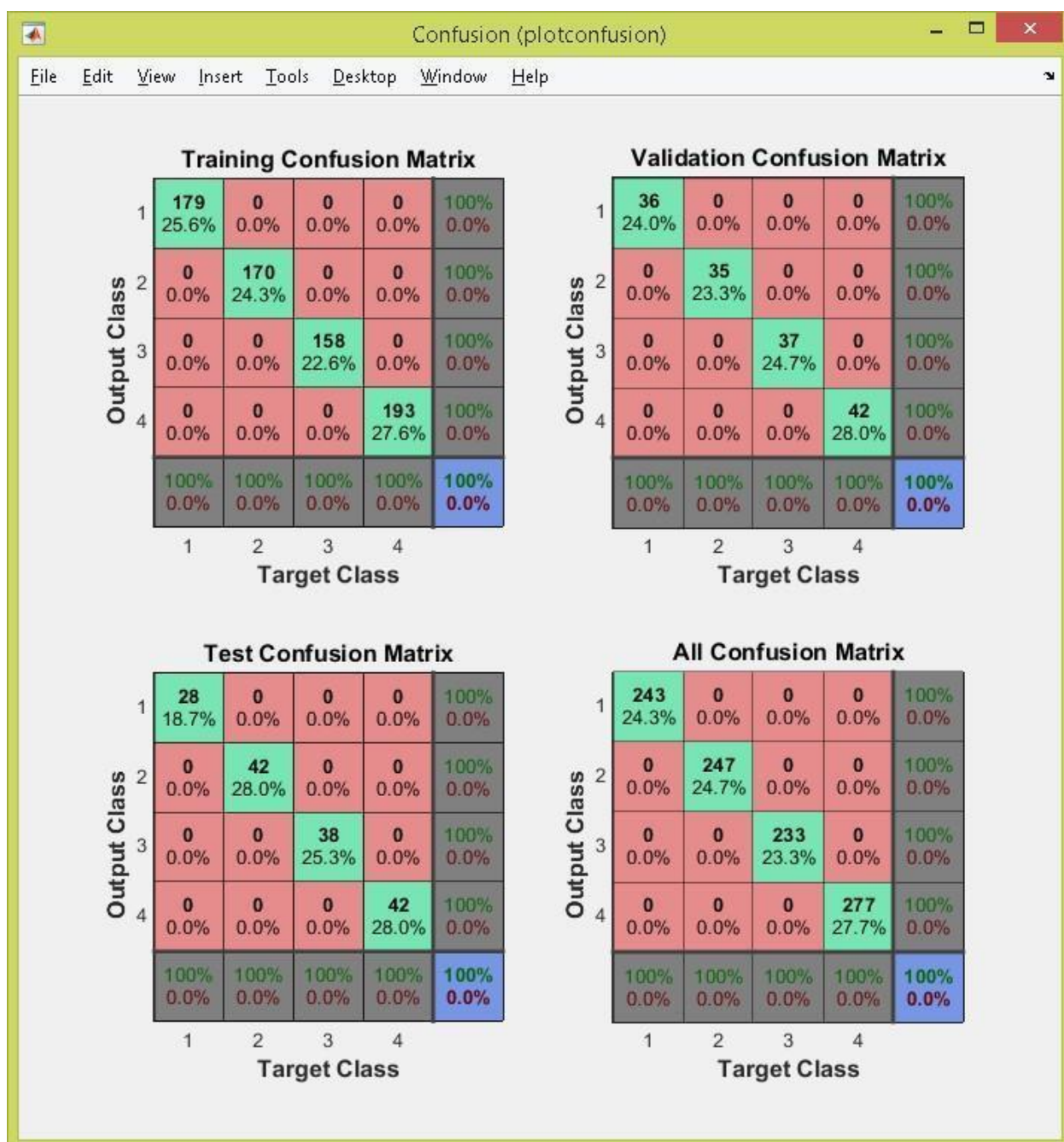


Рисунок 4.4 – График результатов обучения нейронной сети распознавания образов

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ РАБОТЫ

Построить компьютерную модель нейронной сети распознавания образов на основе демонстрационного примера, обучить ее и сохранить полученные результаты в различном виде. Просимулировать сеть исходными данными и сравнить с данными демонстрационных примеров. Рассчитать среднеквадратическую ошибку на всех образах обучающей выборки. Попробуйте изменить архитектуру сети. Сделать вывод о работе сети и качестве исходных данных обучающей выборки.

ОТЧЕТНОСТЬ ЗА ЗАНЯТИЕ

В результате выполненной работы заполняется бланк отчета, в котором должны быть отражены:

1. Тема, программа и цель работы.
2. Задание для лабораторной работы.
3. Структура нейронной сети распознавания образов. Графики погрешности распознавания. Значение ошибки. Результаты симулирования сети.
4. Выводы по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каком образом ставится задача распознавания образов?
2. Какой пакет Matlab используется для моделирования нейронных сетей распознавания образов?

ПРАКТИЧЕСКАЯ РАБОТА 7. Разработка нейросетевого алгоритма классификации цветов ириса

Цель:

1. Закрепить знания, полученные на лекции, посвященной нейросетевым методам распознавания образов.
2. Разработать нейросетевой алгоритм классификации цветов ириса на основе демонстрационного примера.

Порядок и последовательность выполнения работы

1. Создание компьютерной модели нейронной сети классификации цветов ириса.
2. Обучение нейронной сети и исследование ее работы.

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ

1. Создание компьютерной модели нейронной сети классификации цветов ириса.

Рассматриваемый демонстрационный пример позволяет создать нейросетевой алгоритм классификации цветов ириса.

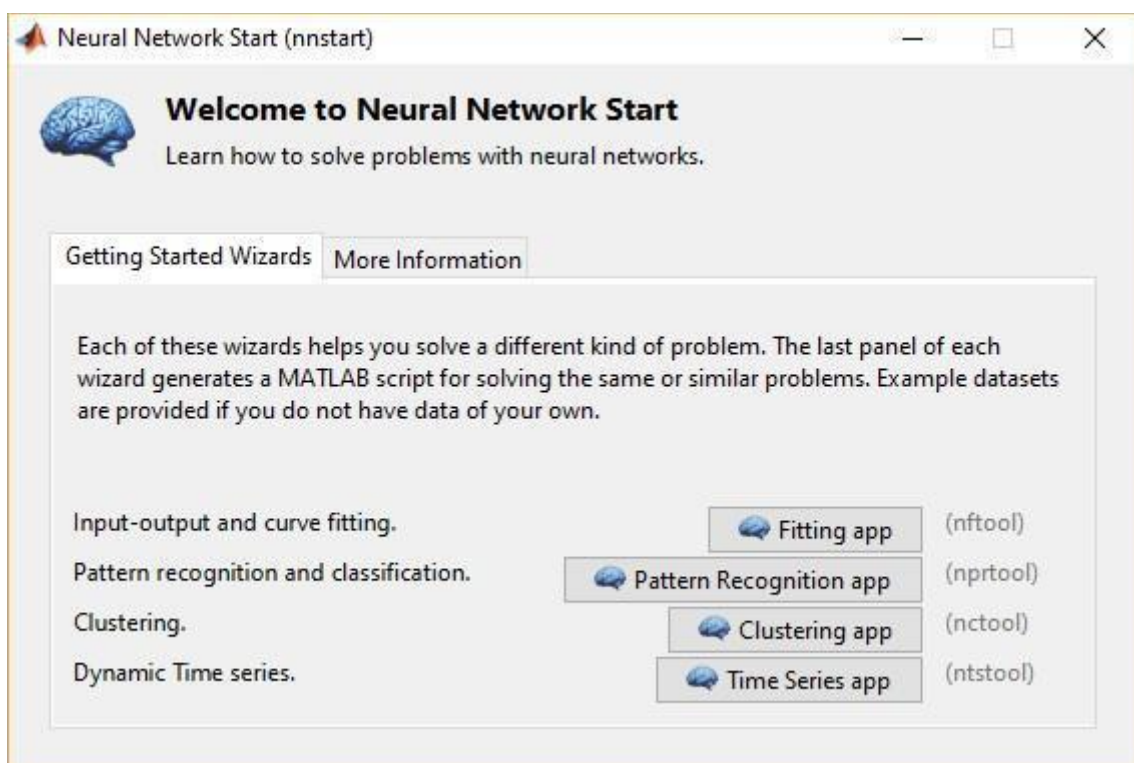


Рисунок 1 – Мастер запуска нейросетевых приложений

Для запуска приложения Neural Pattern Recognition необходимо на



вкладке APPS системного меню выбрать иконку , либо в окне команд набрать

```
>> nprtool,
```

либо запустив окно запуска нейросетевых приложений (рисунок 1)

```
>> nnstart
```

выбрать Pattern Recognition app.

Для загрузки данных этого демонстрационного примера, на основе которого будет создан нейросетевой алгоритм классификации цветов ириса, необходимо выбрать демонстрационный пример (Load Example Data Set) Iris Flowers (рисунок 2) приложения Neural Pattern Recognition.

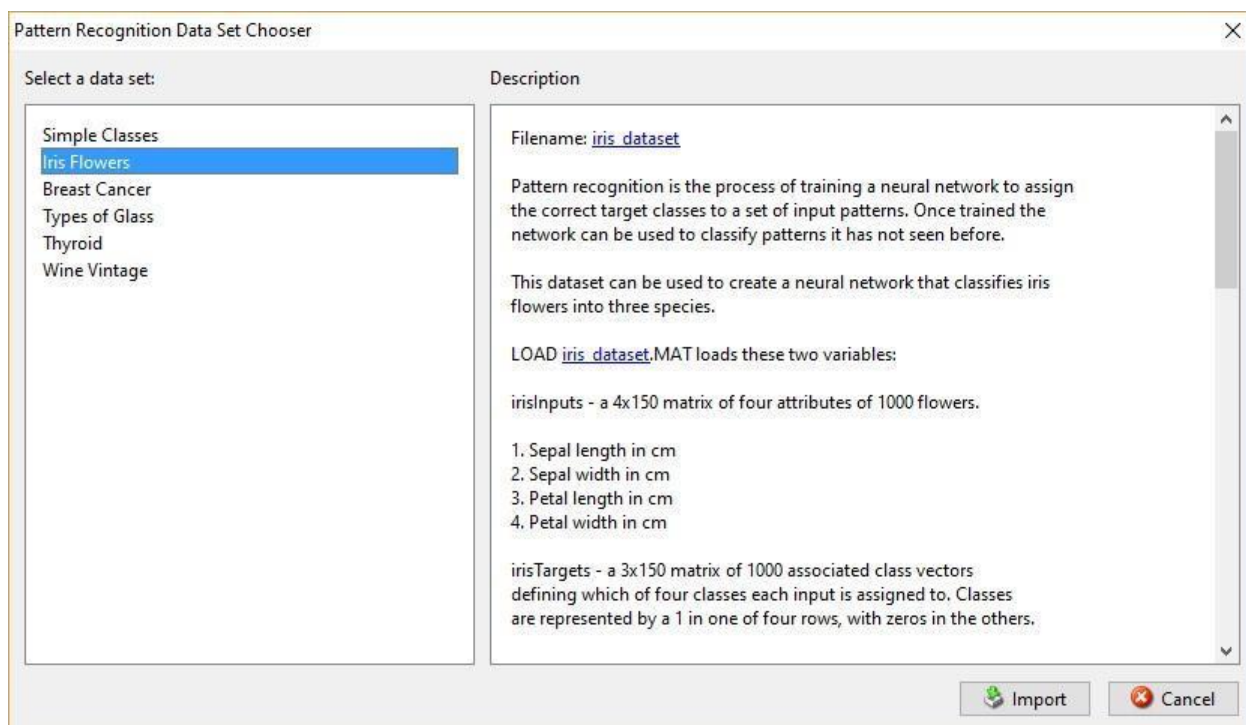


Рисунок 2 – Выбор демонстрационного примера

Исходные данные рассматриваемого демонстрационного примера содержит обучающую выборку из 150 образов, каждый из которых содержит четыре характеристики цветов ириса:

1. Длина чашелистиков (в см),

2. Ширина чашелистиков (в см),
3. Длина лепестка (в см), 4. Ширина лепестка (в см).

Описанные данные содержатся в матрице irisInputs. Фрагмент этой матрицы, содержащий первые 10 образцов обучающей выборки приведен на рисунке 3.

irisInputs										
4x150 double										
	1	2	3	4	5	6	7	8	9	10
1	5.1000	4.9000	4.7000	4.6000	5	5.4000	4.6000	5	4.4000	4.9000
2	3.5000	3	3.2000	3.1000	3.6000	3.9000	3.4000	3.4000	2.9000	3.1000
3	1.4000	1.4000	1.3000	1.5000	1.4000	1.7000	1.4000	1.5000	1.4000	1.5000
4	0.2000	0.2000	0.2000	0.2000	0.2000	0.4000	0.3000	0.2000	0.2000	0.1000

Рисунок 3 – Исходные данные

В качестве выходных данных обучающая выборка содержит цель – принадлежность цветка к одному из трех классов. Описанные данные содержатся в матрице IrisTargets. Матрица содержит три строки. Значения матрицы – «0» или «1», причем «1» обозначает принадлежность цветка к классу, соответствующему номеру строки, в которой находится «1». Фрагмент матрицы IrisTargets приведен на рисунке 4.

irisTargets										
3x150 double										
	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0

Рисунок 4 – Цель

Для создания компьютерной модели нейронной сети распознавания образов необходимо при помощи мастера определить разбиение обучающей выборки (рисунок 5), затем задать архитектуру сети, указав количество нейронов в скрытом слое, по умолчанию – 10 (рисунок 6).

Для обучения сети на следующем шаге работы мастера необходимо нажать кнопку Train (рисунок 7).

Результаты обучения сети отображаются на форме Neural Network Training, представленной на рисунке 8. Видно, что нейронная сеть обучена за 15 итераций.

На следующем шаге работы мастера можно улучшить сеть: переобучив её (Train Again), изменив архитектуру сети (Adjust Network Size) или загрузив данные другой обучающей выборки (Import Larger Data Set). Эти элементы управления находятся в левой

части формы мастера. Кроме это, имеется возможность тестирования работы сети на всех образах обучающей выборки. Для этого на правой стороне формы улучшения сети необходимо выбрать матрицы исходных данных и цели обучающей выборки, и нажать кнопку Test Network.

После что будет активирована кнопка Plot Confusion, при нажатии на которую будет выведено окно с результатами тестирования сети (рисунок 10).

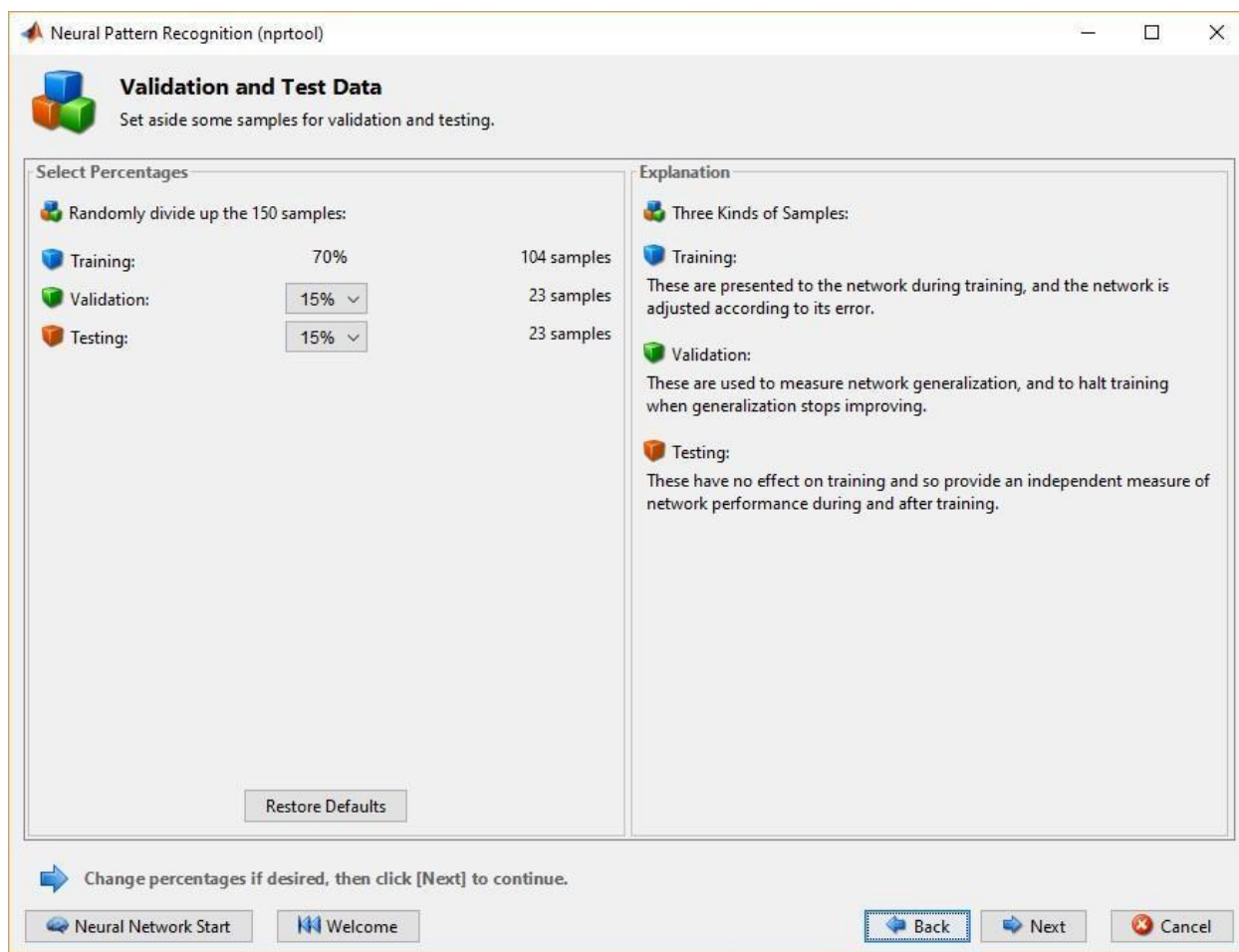


Рисунок 5 – Разбиение данных обучающей выборки

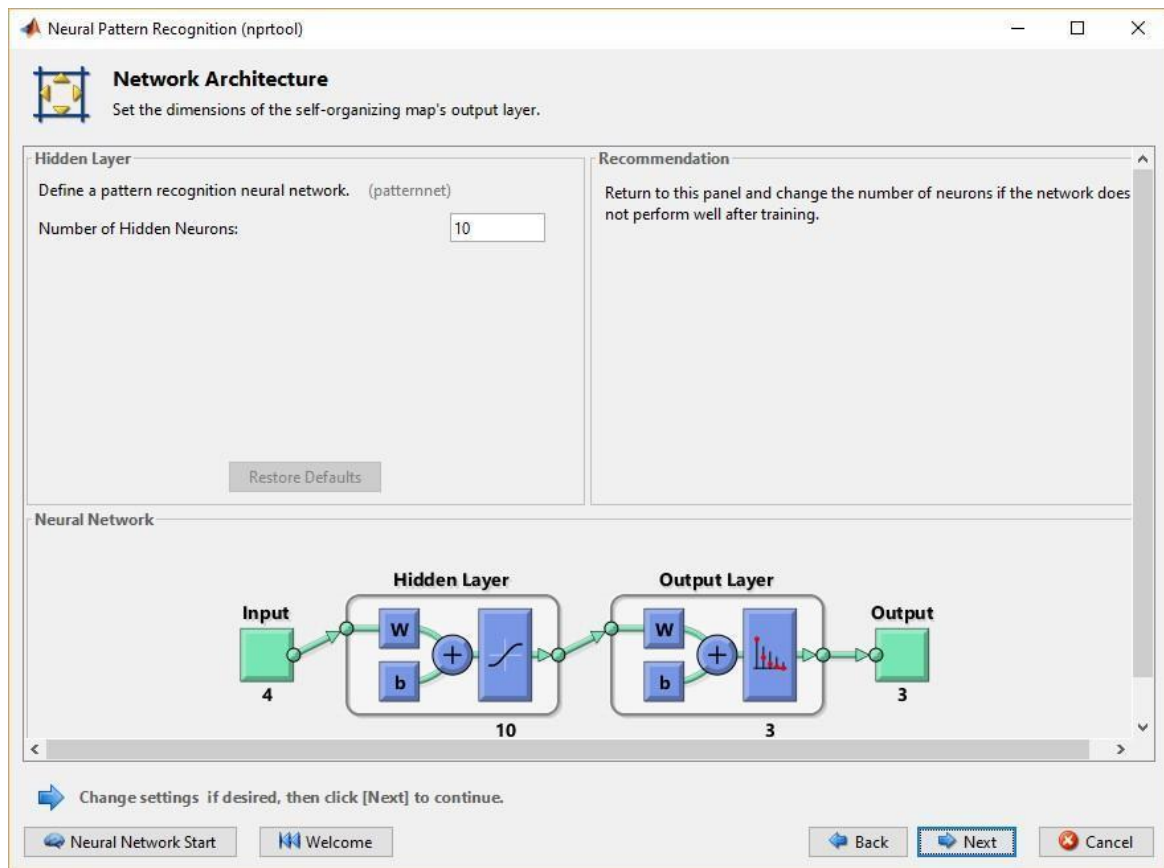


Рисунок 6 – Архитектура нейронной сети

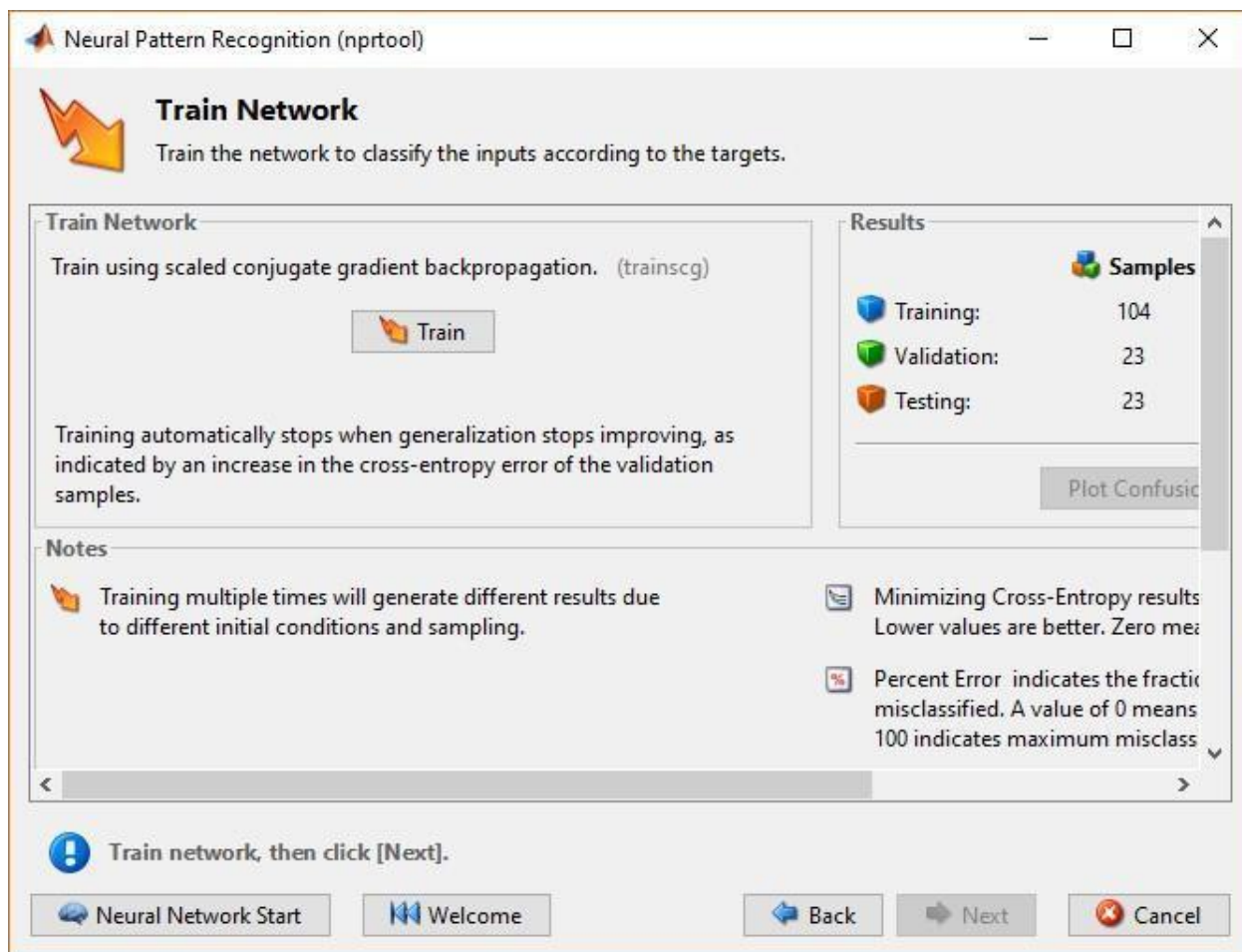


Рисунок 7 – Обучение нейронной сети распознавания образов

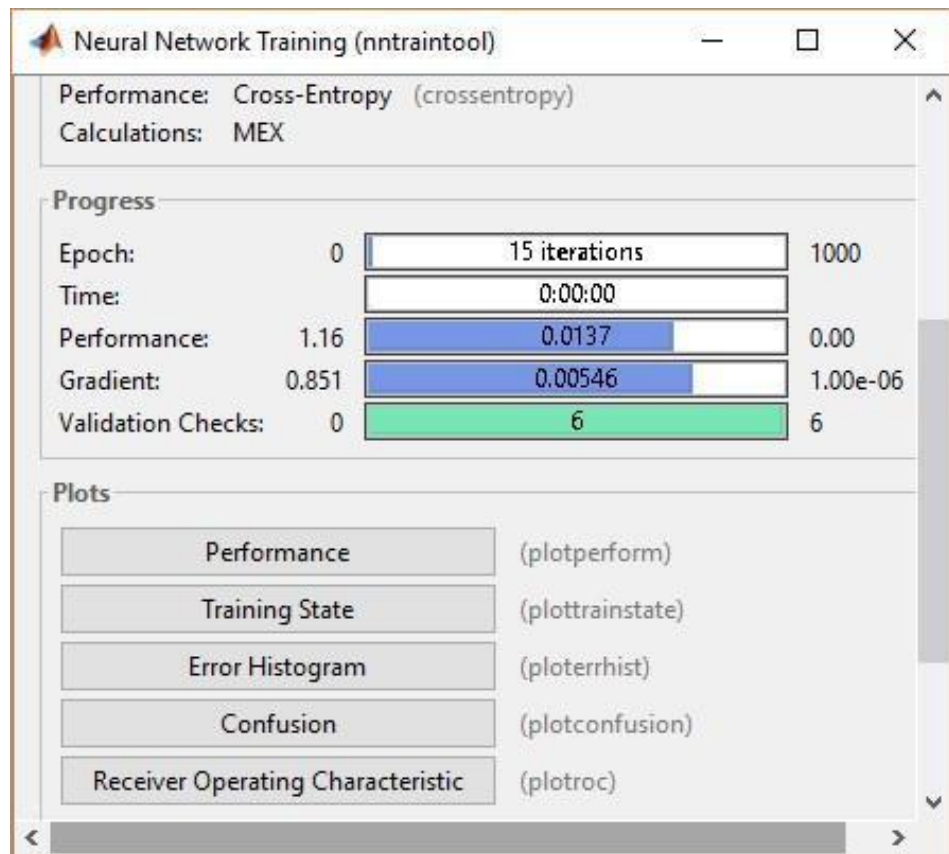


Рисунок 8 – Результаты обучения нейронной сети

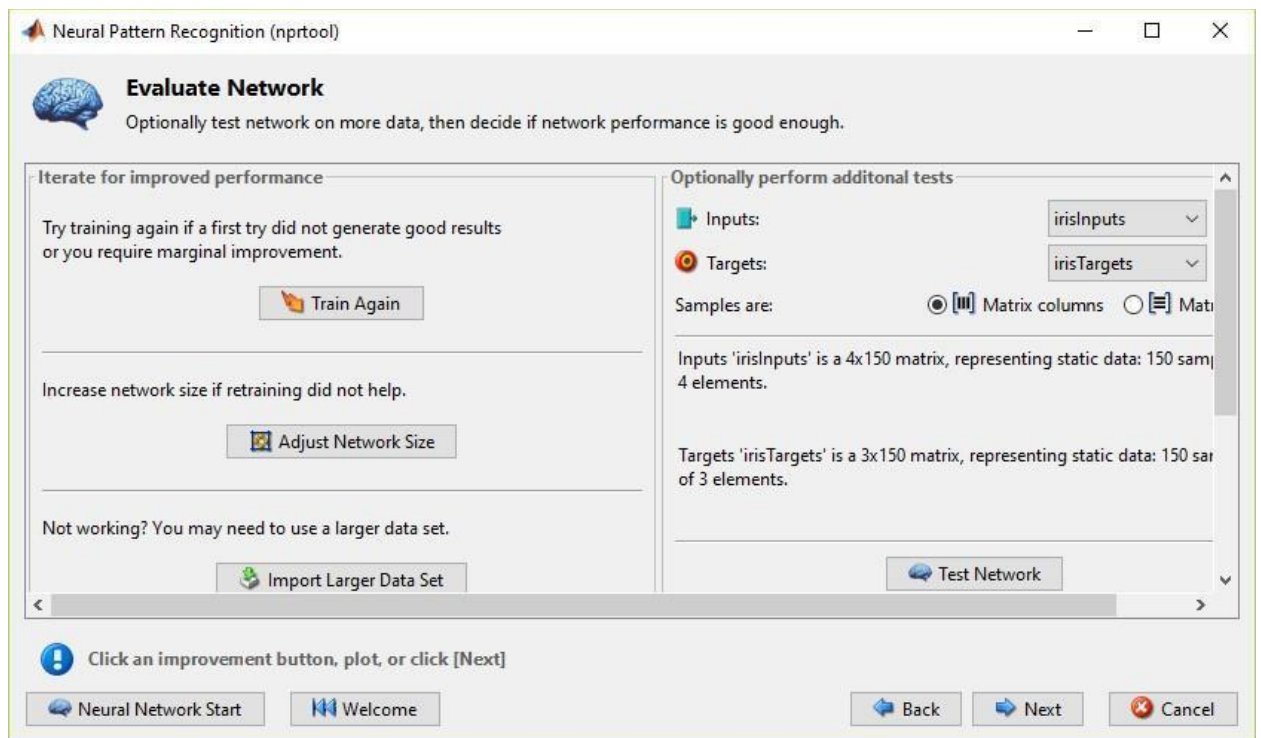


Рисунок 9 – Форма улучшения сети

Confusion Matrix				
Output Class	1	2	3	
	50 33.3%	0 0.0%	0 0.0%	100% 0.0%
	0 0.0%	49 32.7%	1 0.7%	98.0% 2.0%
	0 0.0%	1 0.7%	49 32.7%	98.0% 2.0%
	1	2	3	
Target Class				

Рисунок 10 – Результаты тестирования сети

Анализ полученных результатов показывает, что два образа обучающей выборки распознаны неправильно.

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ РАБОТЫ

Построить компьютерную модель нейронной сети нейросетевой алгоритма классификации цветов ириса на основе демонстрационного примера, обучить ее, протестировать, и сохранить полученные результаты в различном виде. Сделать вывод о работе сети и качестве исходных данных обучающей выборки.

ОТЧЕТНОСТЬ ЗА ЗАНЯТИЕ

В результате выполненной работы оформляется отчет, в котором должны быть отражены:

1. Тема, программа и цель работы.
2. Задание для лабораторной работы.
3. Структура нейронной сети распознавания образов. Результаты тестирования сети.
4. Выводы по работе.
5. Ответы на контрольные вопросы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Каком образом ставится задача распознавания образов?
2. Какой пакет Matlab используется для моделирования нейронных сетей распознавания образов?
3. Какая нейросетевая парадигма используется в задачах распознавания образов?

4. Какой алгоритм обучения нейронной сети используется для распознавания образов?

ПРАКТИЧЕСКАЯ РАБОТА 8. Распознавание букв

Цель занятия:

1. Расширить и углубить знания, полученные на лекциях по основам теории нейронных сетей.
2. Приобрести первичные навыки по проведению исследований и экспериментов с использованием компьютерных моделей информационных систем.

Порядок проведения занятия

1. Создание компьютерной модели нейронной сети.
2. Обучение сети распознаванию образов.
3. Проверка эффективности функционирования сети.

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ

1. Создание компьютерной модели нейронной сети

Часто полезно иметь устройство, которое выполняет распознавание образов. В частности, очень эффективны и выгодны машины, которые могут читать символы. Машина, которая читает банковские чеки, может выполнять за то же самое время намного больше проверок, чем человек. Этот вид приложений сохраняет время и деньги, а также устраняет условия, при которых человек выполняет монотонную, периодически повторяющуюся работу. Демонстрационная программа сценарий `aprcr1` иллюстрирует, как распознавание символов может быть выполнено в сети с обратным распространением.

Постановка задачи

Требуется создать нейронную сеть для распознавания 26 символов латинского алфавита. В качестве датчика предполагается использовать систему распознавания, которая выполняет оцифровку каждого символа, находящегося в поле зрения. В результате каждый символ будет представлен шаблоном размера 5x7. Например, символ А может быть представлен, как это показано на рисунок 1.

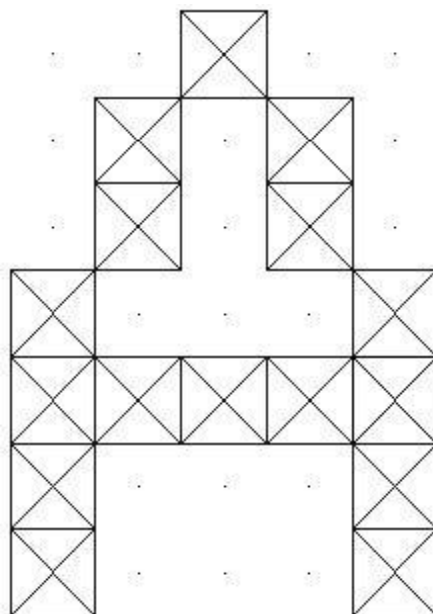


Рисунок 1. – Шаблон символа А

Однако система считывания символов обычно работает неидеально и отдельные элементы символов могут оказаться искаженными (рисунок 2).

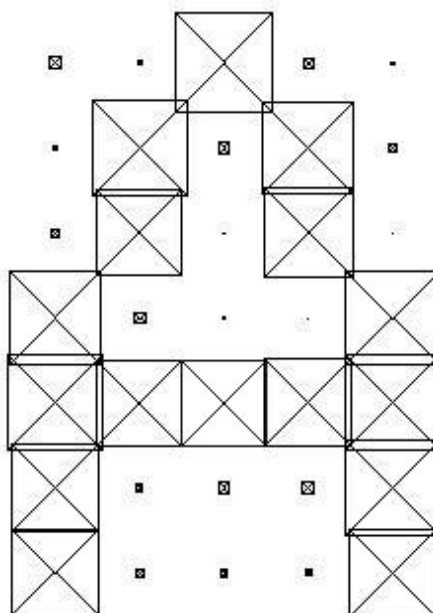


Рисунок 2 – Зашумленный шаблон символа А

Проектируемая нейронная сеть должна точно распознавать идеальные векторы вход и с максимальной точностью воспроизводить зашумленные векторы. М-функция `rgprol` определяет 26 векторов входа, каждый из которых содержит 35 элементов, этот массив называется алфавитом. М-функция формирует выходные переменные `alphabet` и `targets` которые определяют массивы алфавита и целевых векторов. Массив `targets`

определяется как `eye(26)`. Для того чтобы восстановить шаблон для i -й буквы алфавита, надо выполнить следующие операторы:

```
[alphabet, targets] = prprob;
```

```
ti = alphabet(:, i);
```

```
letter{i} = reshape(ti, 5, 7)';
```

```
letter{i}
```

Определим шаблон для символа А, который является первым элементом алфавита

```
[alphabet, targets] = prprob;
```

```
i = 2;
```

```
ti = alphabet(:, i);
```

```
letter{i} = reshape(ti, 5, 7)';
```

```
letter{i}
```

```
ans =
```

```
00100
```

```
01010
```

```
01010
```

```
10001
```

```
11111
```

```
10001
```

```
10001
```

Нейронная сеть

На вход сети поступает вектор входа с 35 элементами; вектор выхода содержит 26 элементов, только один из которых равен 1, а остальные - 0. Правильно

4>ункционирую-щая сеть должна ответить вектором со значением 1 для элемента, соответствующего номеру символа в алфавите. Кроме того, сеть должна быть способной распознавать символы в условиях действия шума. Предполагается, что шум - это случайная величина со средним значением 0 и стандартным отклонением, меньшим или равным 0.2. Архитектура сети

Для работы нейронной сети требуется 35 входов и 26 нейронов в выходном слое. Для рения задачи выберем двухслойную нейронную сеть с логарифмическими сигмоидаль-выми функциями активации в каждом слое. Такая функция активации выбрана потому, то диапазон выходных сигналов для этой функции определен от 0 до 1, и этого достаточно, чтобы сформировать значения выходного вектора.

Скрытый слой имеет 10 нейронов. Такое число нейронов выбрано на основе опыта и разумных предположений. Если при обучении сети возникнут затруднения, то можно

увеличить количество нейронов этого уровня. Сеть обучается так, чтобы сформировать единицу в единственном элементе вектора выхода, позиция которого соответствует номеру символа, и заполнить остальную часть вектора нулями. Однако наличие шумов может приводить к тому, что сеть не будет формировать вектора выхода, состоящего точно из единиц и нулей. Поэтому по завершении этапа обучения выходной сигнал обрабатывается М-функцией `comet`, которая присваивает значение 1 единственному элементу вектора выхода, а всем остальным - значение 0.

Инициализация сети

Вызовем М-файл `prprob`, который формирует массив векторов входа `alphabet` размера 35x26 с шаблонами символов алфавита и массив целевых векторов `targets`:

```
[alphabet,targets] = prprob;
```

```
[R,Q] = size(alphabet);
```

```
[S2,Q] = size(targets);
```

Двухслойная нейронная сеть создается с помощью команды `newff`:

```
S1 = 10;
```

```
net = newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');
```

```
net.LW{2,1} = net.LW{2,1}*0.01;
```

```
net.b{2} = net.b{2}*0.01;
```

Структура нейронной сети, может быть получена при помощи команды `gensim(net)` и представлена на рисунок 3.

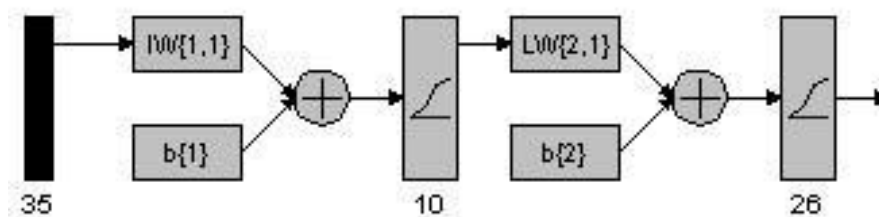


Рисунок 3 – Структура нейронной сети

2. Обучение сети распознаванию образов

Чтобы создать нейронную сеть, которая может обрабатывать зашумленные векторы входа, следует выполнить обучение сети как на идеальных, так и на зашумленных векторах. Сначала сеть обучается на идеальных векторах, пока не будет обеспечена минимальная сумма квадратов погрешностей. Затем сеть обучается на 10 наборах идеальных и зашумленных векторов. Две копии свободного от шума алфавита

используются для того, чтобы сохранить способность сети классифицировать идеальные векторы входа. К сожалению, после того, как описанная выше сеть обучилась классифицировать сильно зашумленные векторы, она потеряла способность правильно классифицировать некоторые векторы, свободные от шума. Следовательно, сеть снова надо обучить на идеальных векторах. Это гарантирует, что сеть будет работать правильно, когда на ее вход будет передан идеальный символ. Обучение выполняется с помощью функции `trainbpx`, которая реализует метод обратного распространения ошибки с возмущением и адаптацией параметра скорости настройки.

Обучение в отсутствие шума

Сеть первоначально обучается в отсутствие шума с максимальным числом циклов обучения 5000 либо до достижения допустимой средней квадратичной погрешности равной 0.1 (рисунок 4);

```
P = alphabet;  
T = targets;  
net.performFcn = 'sse';  
net.trainParam.goal = 0.1;  
net.trainParam.show = 20;  
net.trainParam.epochs = 5000;  
net.trainParam.mc = 0.95;  
[net,tr] = train(net,P,T);
```

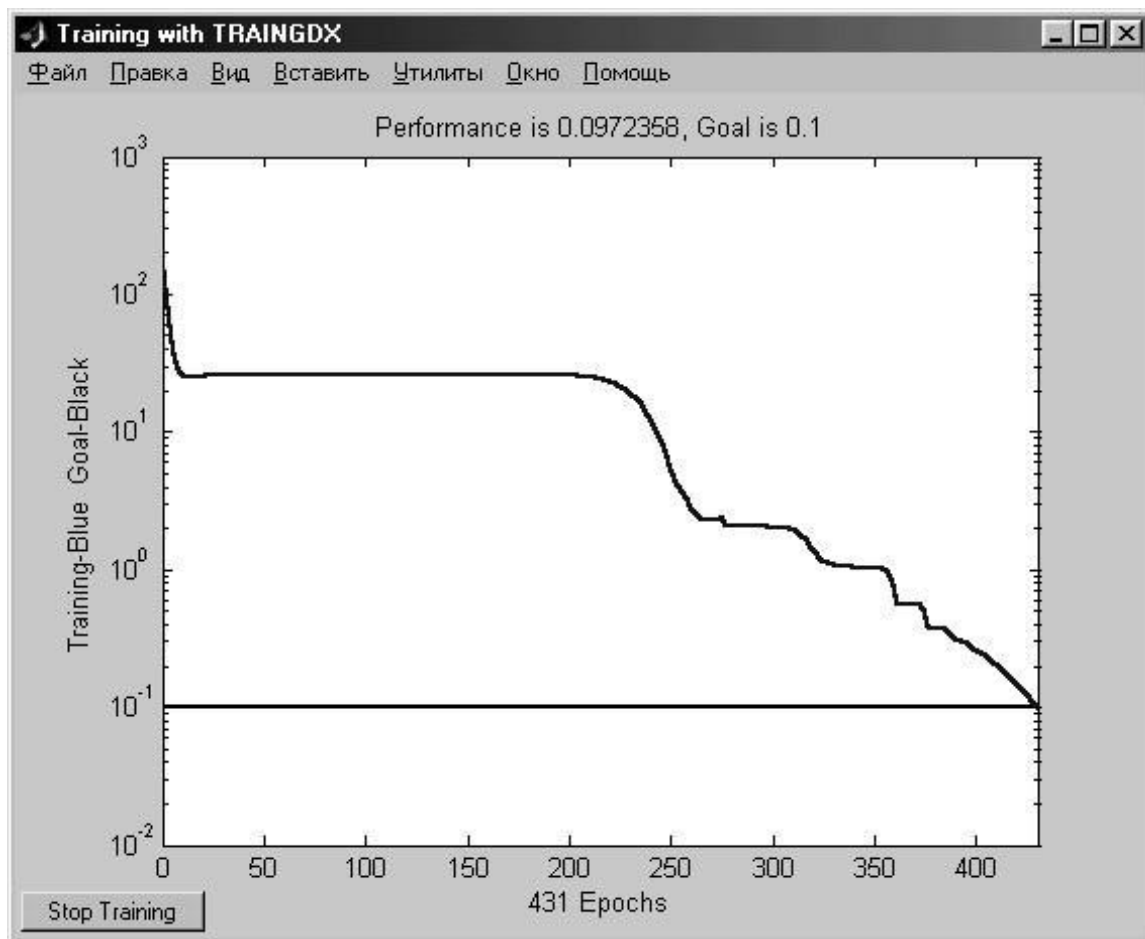


Рисунок 4 – Обучение нейронной сети в отсутствии шума

Обучение в присутствии шума

Чтобы спроектировать нейронную сеть, не чувствительную к воздействию шума, обучим ее с применением двух идеальных и двух зашумленных копий векторов алфавита. Целевые векторы состоят из четырех копий векторов. Зашумленные векторы имеют шум со средним значением 0.1 и 0.2. Это обучает нейрон правильно распознавать зашумленные символы и в то же время хорошо распознавать идеальные векторы.

При обучении с шумом максимальное число циклов обучения сократим до 300, а допустимую погрешность увеличим до 0.6 (рисунок 5):

```
netn = net;
netn.trainParam.goal = 0.6;
netn.trainParam.epochs = 300;
T = [targets targets targets targets]; for pass = 1:10
P = [alphabet, alphabet, ...
    (alphabet + randn(R,Q)*0.1), ...
```

```

(alphabet + randn(R,Q)*0.2)];

[netn,tr] = train(netn,P,T);

end

```

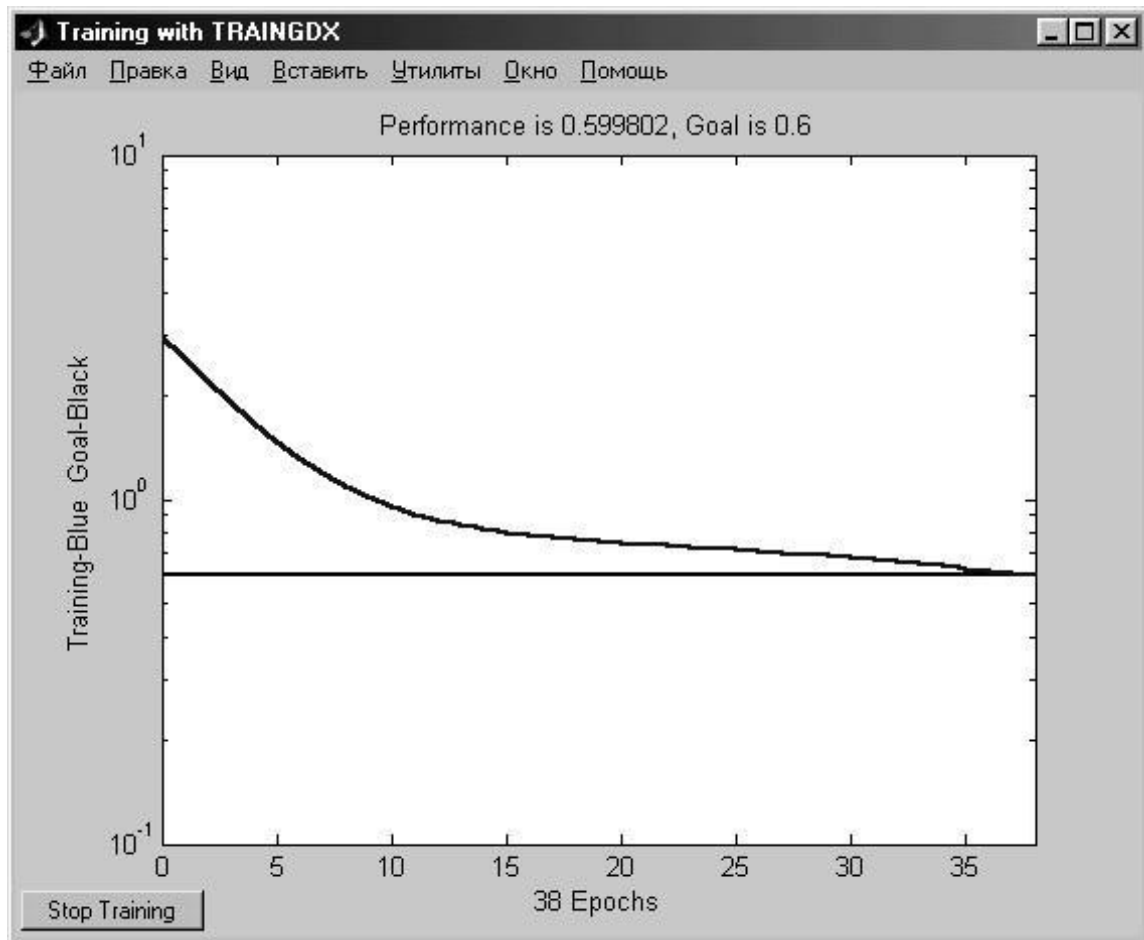


Рисунок 5 – Обучение сети с шумом

Повторное обучение в отсутствие шума

Поскольку нейронная сеть обучалась в присутствии шума, то имеет смысл повторить ее обучение без шума, чтобы гарантировать, что идеальные векторы входа классифицируются правильно.

```

netn.trainParam.goal = 0.1;

netn.trainParam.epochs = 500;

net.trainParam.show =5;

[netn, tr] = train(netn, P, T);

```

3. Проверка эффективности функционирования сети

Эффективность нейронной сети будем оценивать следующим образом.

Рассмотрим 2 структуры нейронной сети: сеть 1, обученную на идеальных последовательностях, и сеть 2, обученную на зашумленных последовательностях. Проверка функционирования производится на 100 векторах входа при различных уровнях шума. Приведем фрагмент сценария, который выполняет эти операции:

```
noise_range = 0:.05:.5;
max_test = 100;
network1 = [];
network2 = [];
T = targets;
for noiselevel = noise_range
    fprintf('Уровень шума %.2f.\n',noiselevel);
    errors1 = 0;
    errors2 = 0;
    for i=1:max_test
        P = alphabet + randn(35,26)*noiselevel;
        % Тест для сети 1
        A = sim(net,P);  AA = compet(A);
        errors1 = errors1 + sum(sum(abs(AA-T)))/2;
        % Тест для сети 2
        An = sim(netn,P);
        AAn = compet(An);
        errors2 = errors2 + sum(sum(abs(AAn-T)))/2;
    end
    network1 = [network1 errors1/26/100];
    network2 = [network2 errors2/26/100];
end
plot(noise_range,network1*100,'--',noise_range,network2*100);
title('Процент ошибочных классификаций');
xlabel('Уровень шума');
ylabel('Сеть 1 - - Сеть 2 ---');
```

Тестирование реализуется следующим образом. Шум со средним значением 0 и стандартным отклонением от 0 до 0.5 с шагом 0.05 добавляется к векторам входа. Для каждого уровня шума формируется 100 зашумленных последовательностей для каждого символа вычисляется выход сети. Выходной сигнал обрабатывается М-функцией `compnet` с той целью, чтобы выбрать только один из 26 элементов вектора выхода. После этого оценивался количество ошибочных классификаций и вычисляется процент ошибки. Соответствующий график погрешности сети от уровня входного шума показан на рисунке 6.

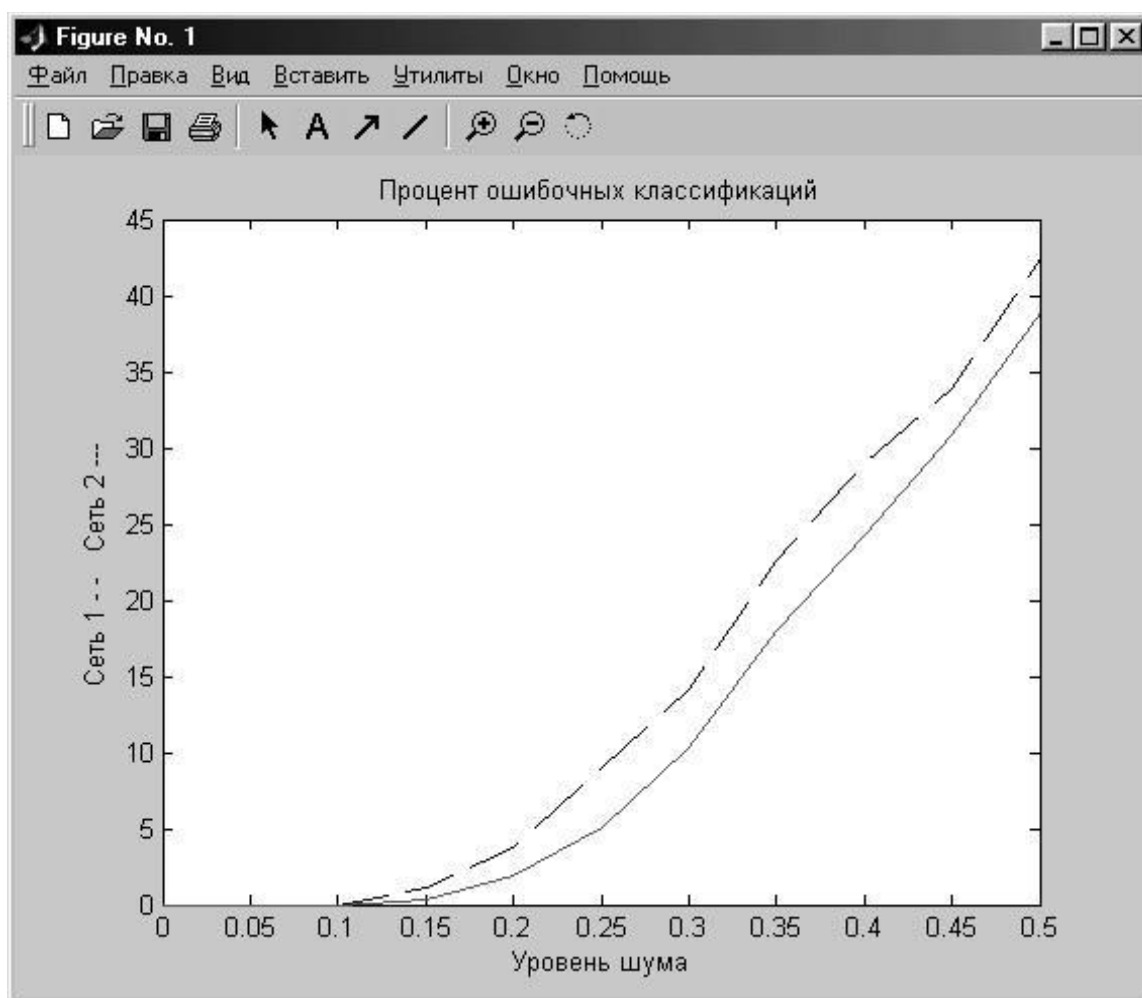


Рисунок 6 – График сравнения эффективности распознавания образов нейронными сетями, обученных с шумом и без

Сеть 1 обучена на идеальных векторах входа, а сеть 2 $\square$ на зашумленных. Обучение сети на зашумленных векторах входа значительно снижает погрешность распознавания реальных векторов входа. Сети имеют очень малые погрешности, если среднеквадратичное значение шума находится в пределах от 0.00 до 0.05. Когда к векторам был добавлен шум со среднеквадратичным значением 0.2, в обеих сетях начали возникать заметные ошибки. При этом погрешности нейронной сети, обученной на

зашумленных векторах, на 1-4 % ниже, чем для сети, обученной на идеальных входных последовательностях.

Если необходима более высокая точность распознавания, сеть может быть обучена либо в течение более длительного времени, либо с использованием большего количества нейронов в скрытом слое. Можно также увеличить размер векторов, чтобы пользоваться шаблоном с более мелкой сеткой, например 10x14 точек вместо 5x7.

Проверим работу нейронной сети для распознавания символов. Сформируем зашумленный вектор входа для символа J (рисунок 7):

```
noisyJ = alphabet(:,10) + randn(35,1) * 0.2;  
plotchar(noisyJ); % Зашумленный символ J (рис. 7)
```

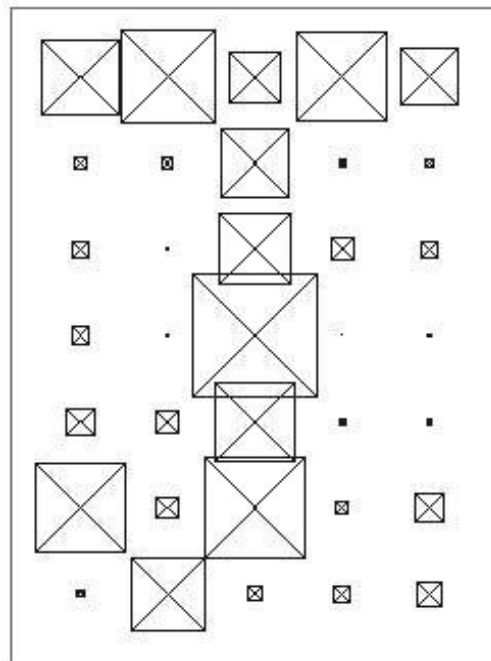


Рисунок 7 – Зашумленный символ J

```
A2 = sim(net,noisyJ);  
A2 = compet(A2);  
answer = find(compet(A2) == 1) answer = 10  
plotchar(alphabet(:,answer)); % Распознанный символ J
```

Нейронная сеть выделила сигнал на 10-м выходном нейроне и восстановила символ J без ошибок (рисунок 8).

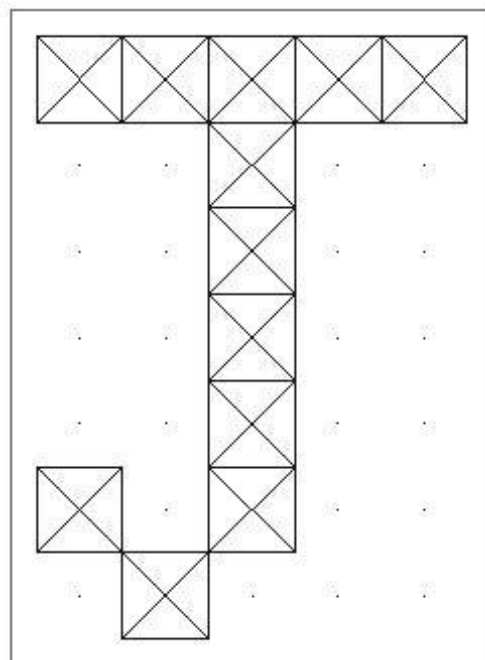


Рисунок 8 – Распознанный символ J

Эта задача демонстрирует, как может быть разработана простая система распознавания изображений. Заметим, что процесс обучения не состоял из единственного обращения к обучающей функции. Сеть была обучена несколько раз при различных векторах входа. Обучение сети на различных наборах зашумленных векторов позволило обучить сеть работать с изображениями, искаженными шумами, что характерно для реальной практики.

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

При помощи пакета Neural Network ToolBox создайте модель нейронной сети распознавания букв английского алфавита. Обучите нейронную сеть распознаванию образов в условиях шума и без него. Исследуйте правильность распознавания образов этими сетями при различных уровнях шума. Сравните эффективность распознавания букв этими сетями сделайте выводы.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какая модель нейронной сети создана на занятии?
2. Чем обусловлена разница эффективности распознавания сетей, обученных с шумом и без него?

ПРАКТИЧЕСКАЯ РАБОТА 9. Использование нейронной сети для приближения функций

Порядок и последовательность выполнения работы

1. Создание компьютерной модели нейронной сети приближения функции.
2. Обучение и моделирование сети. Получение погрешности приближения.

МЕТОДИЧЕСКИЕ ПОЯСНЕНИЯ И РЕКОМЕНДАЦИИ

Пусть необходимо аппроксимировать полином вида

$$Y = x_1^2 + x_2^2 + x_1x_2$$

Для формирования вектора обучающих примеров в окне команд или в m-файле необходимо набрать следующие команды:

```
i=1;  
for x1=-1:.1:1;  
for x2=-1:.1:1;  
D(1,i)=x1;  
D(2,i)=x2;  
T(i)=x1^2-x2^2+x1*x2;  
i=i+1;  
end;  
end;
```

После чего в рабочей области будут сформированы матрица входных данных D и вектор целей T. Импортируем данные из рабочей области: вектор целей T и матрицу входных данных D (рисунок 1).

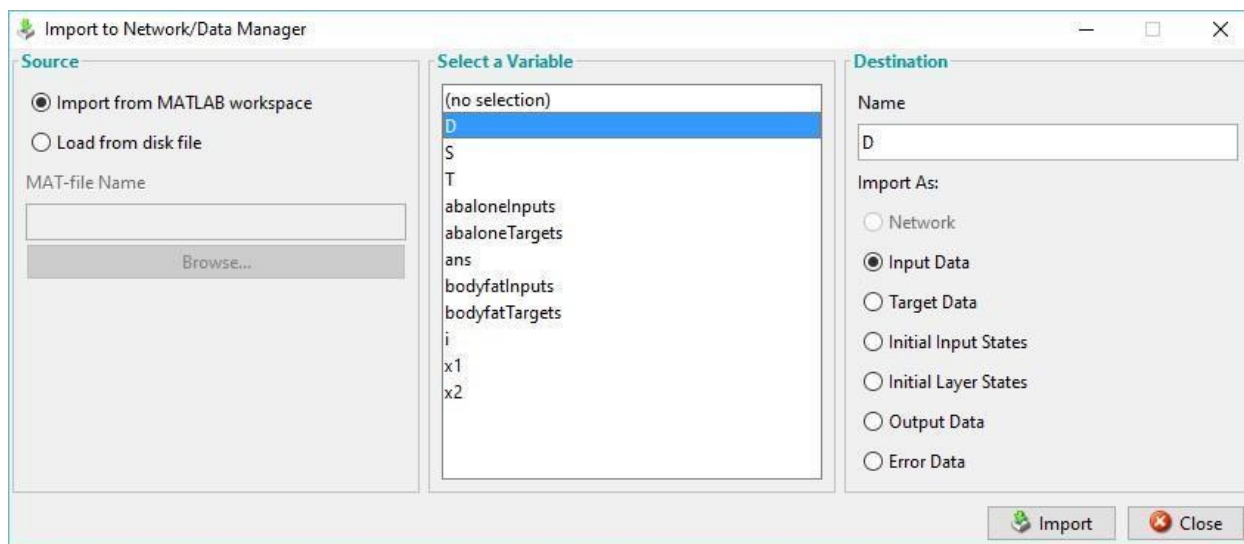


Рисунок 1 – Импорт данных в пакет моделирования нейронных сетей

Построим при помощи инструментального средства Neural Network Tools Network/Data Manager модель нейронной сети. Для этого необходимо выбрать сеть с обратным распространением ошибки (Feed-forward backprop), параметры изменения входных сигналов необходимо выбрать из матрицы D, количество нейронов в первом слое 5, во втором 1, функция активации нейронов первого слоя – любая нелинейная (например логистическая – LOGSIG), второго – линейная (PURELIN). На рисунке 2 представлено окно Network/Data Manager с выбором параметров компьютерной модели создаваемой нейронной сети.

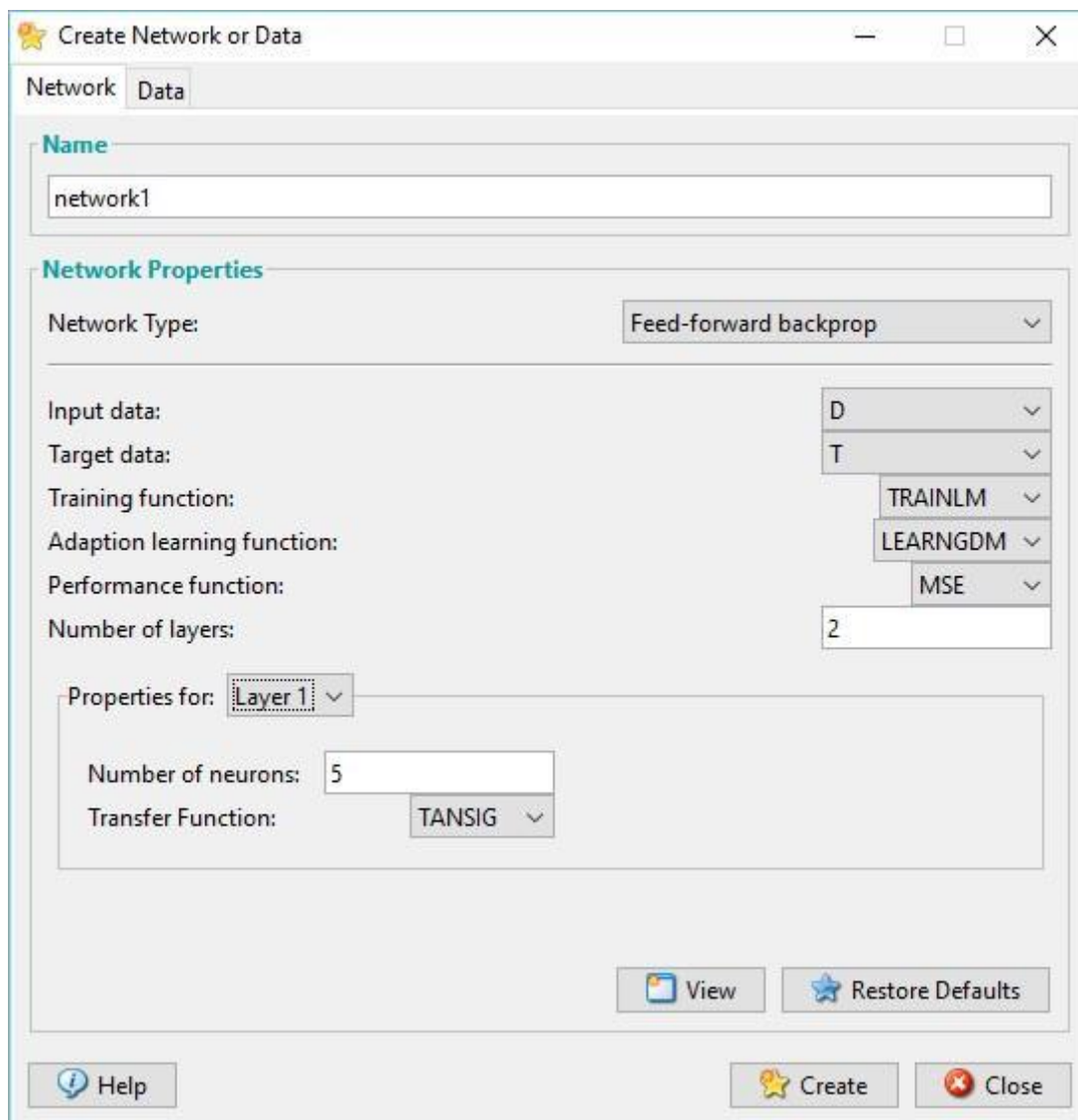


Рисунок 2 – Окно Network/Data Manager с выбором параметров компьютерной модели создаваемой нейронной сети приближения функции

Полученная сеть в среде Simulink представлена на рисунке 3.

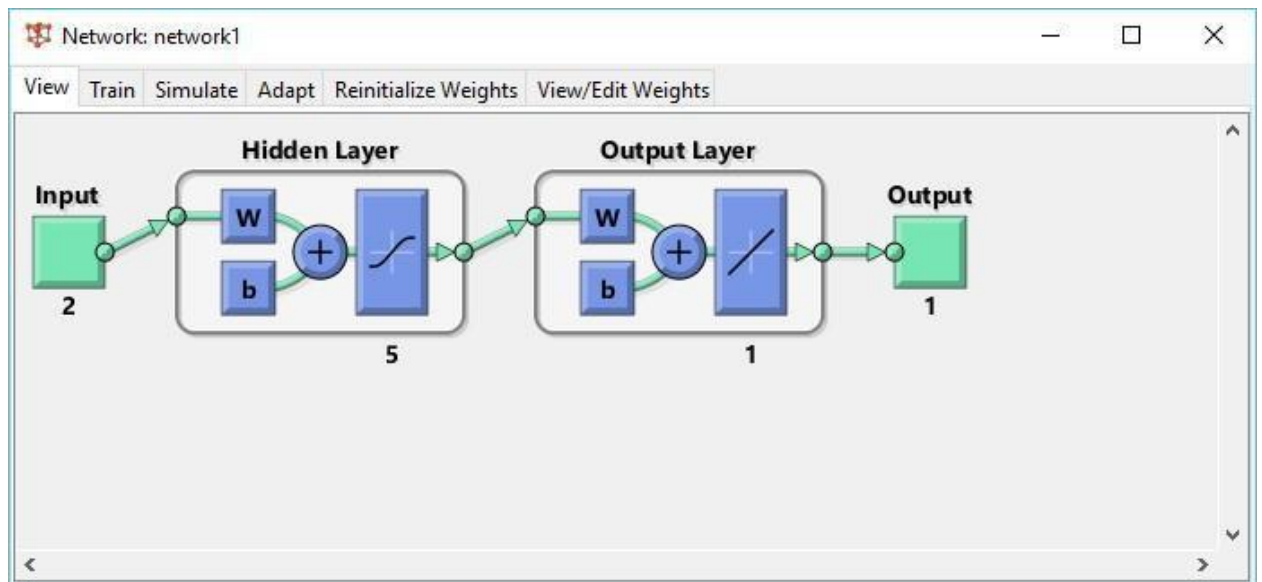


Рисунок 3 – Структура созданной нейронной сети

Обучим сеть, при этом в качестве входных данных необходимо использовать матрицу D, а в качестве цели вектор T (рисунок 4).

The screenshot shows the "Train" tab of the "Network: network1" window. It displays a table of training parameters under the "Training Parameters" sub-tab. The parameters are organized into two columns.

showWindow	true	mu	0.001
showCommandLine	false	mu_dec	0.1
show	25	mu_inc	10
epochs	1000	mu_max	10000000000
time	Inf		
goal	0		
min_grad	1e-05		
max_fail	6		

At the bottom right, there is a button labeled "Train Network" with a small icon.

Рисунок 4 – Задание параметров обучения нейронной сети

Процесс обучения нейронной сети представлен на рисунке 5. Сеть обучена за 98 итераций до ошибки $9.7 \cdot 10^{-6}$.



Рисунок 5 – Процесс обучения нейронной сети

Обученную сеть симулируем вектором данных, получив выход сети `network1_outputs` на основе входных данных матрицы `D` (рисунок 6).

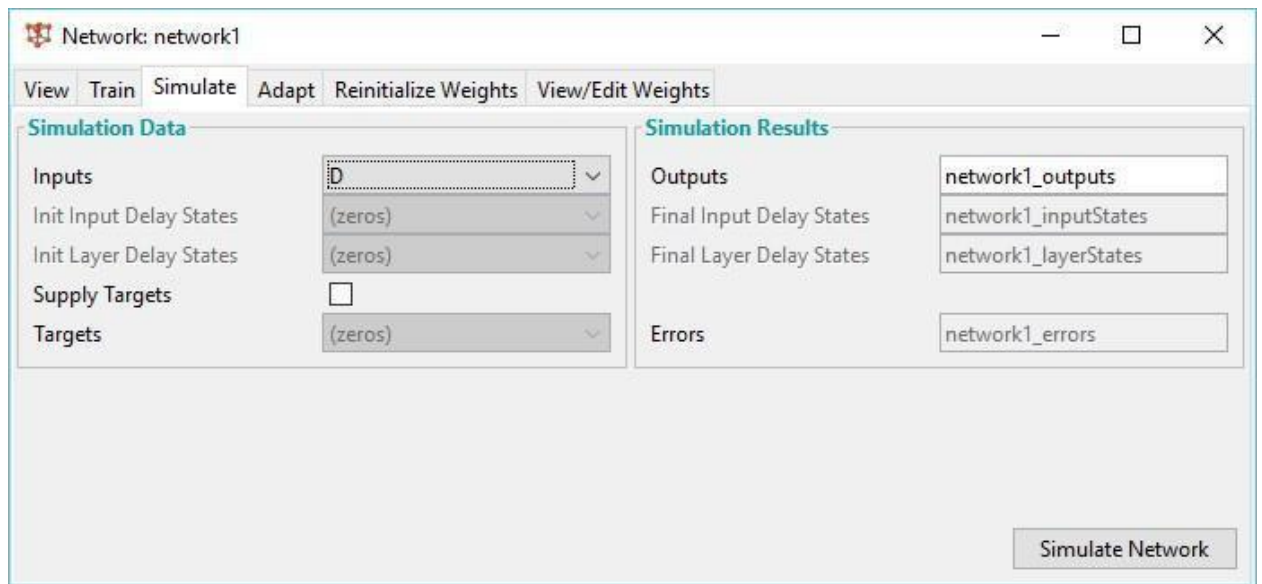


Рисунок 6 – Симулирование нейронной сети

После этого необходимо экспортировать вектор `network1_outputs` в рабочую область системы Matlab (рисунок 7).

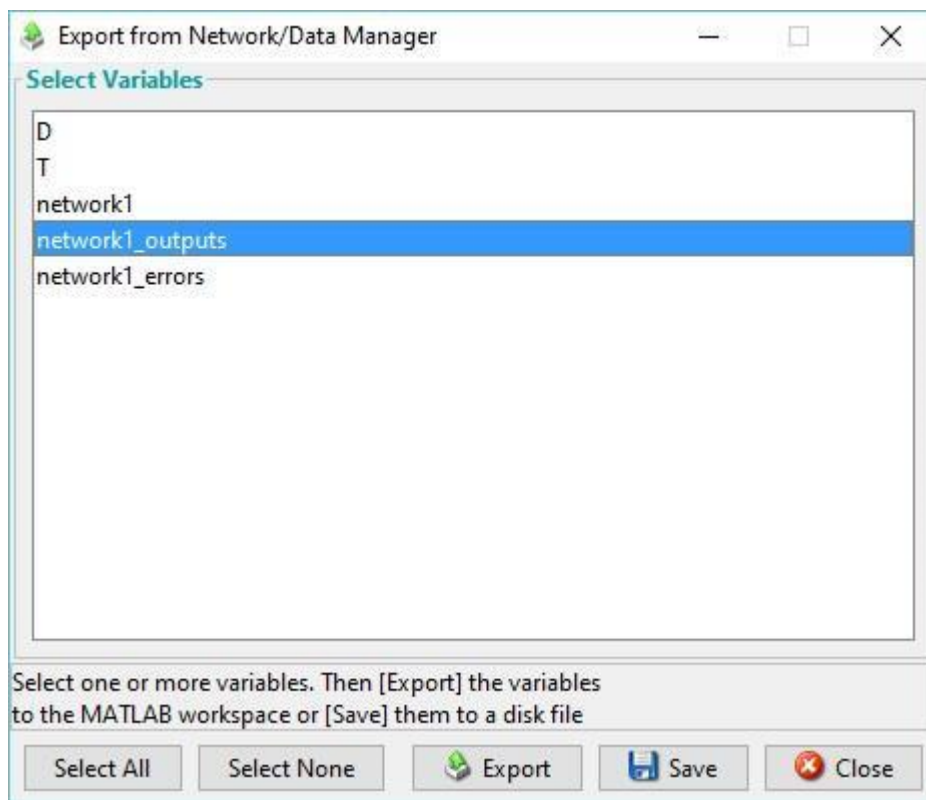


Рисунок 7 – Экспорт данных из пакета NNTool в рабочую область системы

Matlab

Абсолютную ошибку сети можно найти, получив разность выхода обученной сети при симуляции `network1_outputs` и цели `T` в рабочей области: $\text{error} = \text{network1_outputs} - T$; График ошибки представлен на рисунке

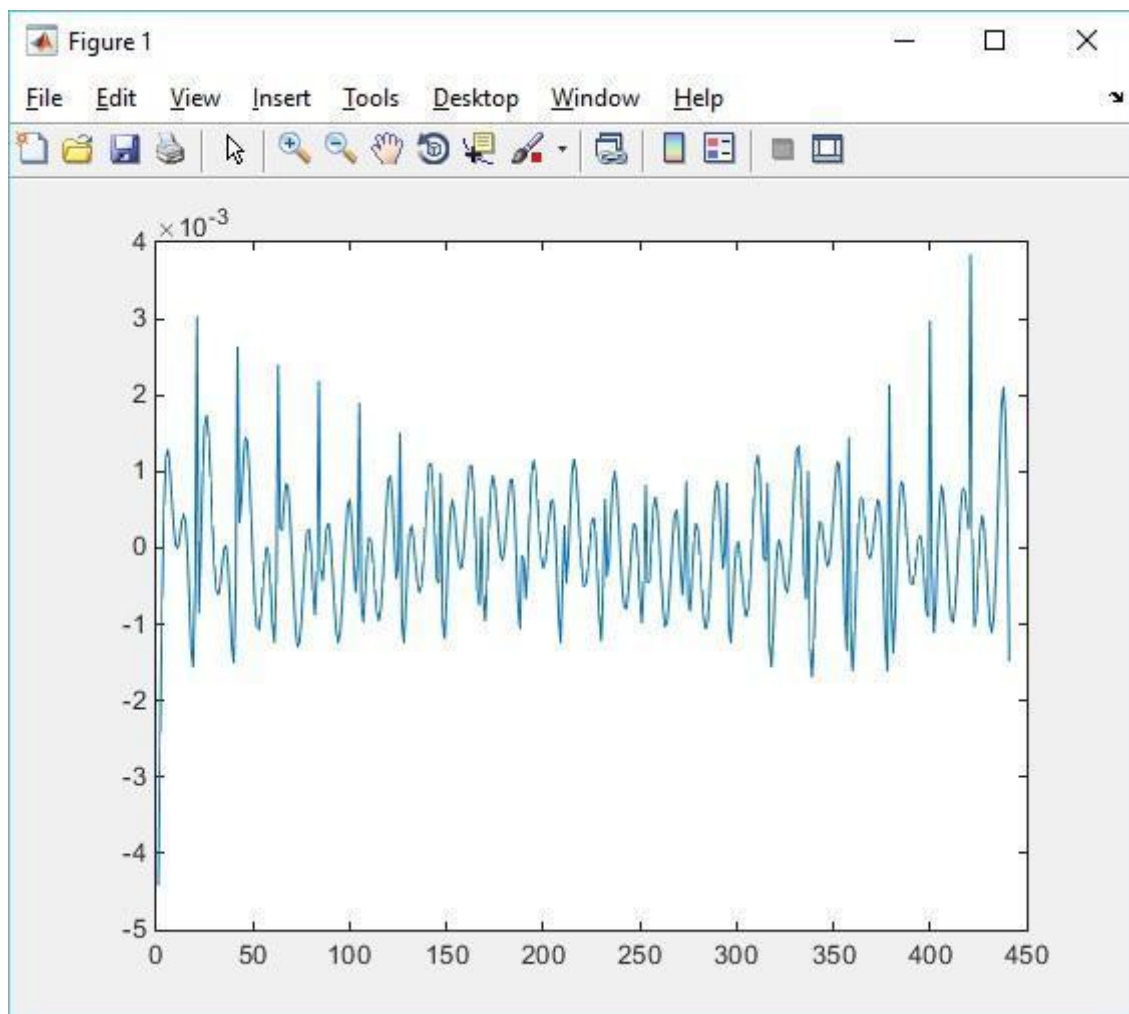


Рисунок 8 – Погрешность приближения функции нейронной сетью

ОТЧЕТНОСТЬ ЗА ЗАНЯТИЕ

В результате выполненной работы заполняется бланк отчета, в котором должны быть отражены:

5. Программа и цель работы.
6. Задание для лабораторной работы по каждому пункту.
7. Структура нейронной сети приближения функции. Графики погрешности приближения функции.
8. Выводы по работе.

ЗАДАНИЕ НА ВЫПОЛНЕНИЕ РАБОТЫ

Построить модель нейронной сети, аппроксимирующей полином вида

$$Y = ax_1^2 + bx_2^2 + cx_1x_2 + dx_1 + ex_2 + f$$

на интервале $[-1, 1]$, коэффициенты полинома заданы в таблице 1. Вариант соответствует порядковому номеру по журналу.

Таблица 1 – Варианты заданий

№ варианта	a	b	c	d	e	f
1.	-1	1	2	0	0	0
2.	1	1	1	1	1	1
3.	2	1	1	2	2	-8
4.	0	1	0	1	0	-2
5.	1	0	1	0	1	-3
6.	2	1	-2	1	0	-7
7.	1	2	3	-3	-4	-1
8.	2	1	2	-1	-3	-2
9.	2	1	1	-2	-2	-1
10	.-1	-1	-1	-1	2	2
11	.-2	1	-1	2	1	-2
12	.-1	3	-1	2	-1	-1
13	.-2	-2	2	2	-2	2
14	.-3	2	-2	3	-1	1
15	.-2	2	-2	2	-2	2
16	.-1	-1	-1	1	1	1
17	.0	1	0	2	0	-3
18	.2	0	2	3	1	0
19	.2	-2	4	-4	3	-3
20	.-5	-3	3	3	2	1
21	.-2	-3	2	3	0	-1
22	.2	6	1	1	-3	-4
23	.3	2	1	2	1	-2
24	.-2	-1	-3	2	3	2
25	.1	1	2	2	-3	-3
26	.-2	-3	-4	4	3	2
27	.2	5	-5	-2	1	-1

Вычертить в отчете структуру нейронной сети, построить графики аппроксимируемой функции, выход нейронной сети и функции погрешности

приближения. Уменьшить шаг дискретизации при задании обучающих примеров вдвое, построить для этого случая функцию ошибок. В каких точках функция ошибки сети равна нулю? Сделайте выводы по полученным результатам.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Постановка задачи обучения нейронной сети.
3. Алгоритм обучения нейронной сети на основе обратного распространения ошибки.
4. Обучение нейронной сети на основе самоорганизации.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

для обучающихся по организации и проведению самостоятельной работы
по дисциплине «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ КОМАНДНОЙ
РАБОТЫ И ПРОЕКТНОЙ ДЕЯТЕЛЬНОСТИ»
для студентов направления подготовки

направленность (профиль):

Ставрополь

2026

СОДЕРЖАНИЕ

1. Общие	положения	29
2. Цель и задачи самостоятельной	работы	30
3. Технологическая карта самостоятельной	работы студента	30
4. Контрольные точки и виды отчетности по ним		31
5. Порядок выполнения самостоятельной работы студентом		33
5.1. Методические рекомендации по работе с учебной литературой		33
5.2. Методические рекомендации по составлению конспекта лекций		7
5.3. Методические рекомендации по подготовке к практическим занятиям		35
5.4. Методические рекомендации по самопроверке знаний		35
5.5. Методические рекомендации по созданию проекта (докладов, рефератов, эссе, научных статей и т.д.)		36
5.6. Методические рекомендации по подготовке к зачетам		41
6. Список литературы для выполнения	СРС	41

1. Общие положения

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов). Владиславу Вячеславовичу

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

- формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);
- написание докладов;
- подготовка к семинарам, практическим и лабораторным работам, их оформление;
- составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);
- выполнение учебно-исследовательских работ, проектная деятельность;
- подготовка практических разработок и рекомендаций по решению проблемной ситуации;
- выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;
- компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;
- выполнение курсовых работ (проектов) в рамках дисциплин;

- выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине «Информационные технологии командной работы и проектной деятельности» направлена на формирование следующих **компетенций**:

Код	Формулировка
ОПК-2	Способен создавать и исследовать новые математические модели в естественных науках, совершенствовать и разрабатывать концепции, теории и методы

2. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование набора общенаучных, профессиональных и универсальных компетенций будущего бакалавра.

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов

способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельной работы и лабораторных занятий.

3. Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций	Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Объем часов, в том числе (астр.)		
				СРС	Контактная работа с преподавателем	Всего
2 семестр						
ОПК-2	Подготовка к лекциям	Конспект лекций	Собеседование	20,0	8,0	28,0
ОПК-2	Подготовка к практическим занятиям	Отчет в электронном виде	Собеседование	46,0	22,0	68,0
ОПК-2	Самостоятельное изучение литературы	Конспект	Собеседование	30,0	12,0	42,0
ОПК-2	Подготовка доклада	Доклад в форме презентации	Доклад	10,0	4,0	14,0
ОПК-2	Самотестирование, подготовка к тестированию	Тестирование	Тестовые задания	20,0	8,0	28,0
Итого за 2 семестр				126,0	54,0	180,0
Итого				126,0	54,0	180,0

4. Контрольные точки и виды отчетности по ним

Контроль качества и сроков самостоятельной работы выполняется в соответствии с учебным графиком и оформляется в соответствии с заданием. Предусмотрена следующая рейтинговая оценка знаний

Описание шкалы оценивания

В рамках рейтинговой системы успеваемость студентов по каждой дисциплине оценивается в ходе текущего контроля и промежуточной аттестации.

Текущий контроль

5. Порядок выполнения самостоятельной работы студентом

5.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист

помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют **четыре основные установки в чтении научного текста:**

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

5.2. Методические рекомендации по составлению конспекта лекций:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта.
2. Выделите главное, составьте план.
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора.

4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.

5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

5.3. Методические рекомендации по подготовке к практическим занятиям

Для того чтобы практические занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на лабораторных занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

Практические работы № 1-5,8 выполняются студентами заочной и очно-заочной формы обучения самостоятельно.

5.4. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала – умение отвечать на вопросы для собеседования.

ВОПРОСЫ ДЛЯ СОБЕСЕДОВАНИЯ

по дисциплине «Искусственные нейронные сети».

2 семестр

1. Какая нейронная сеть называется рекуррентной?
2. Объясните особенности функционирования нейронной сети прямого распространения.
3. Напишите уравнения, характеризующие динамику рекуррентной нейронной сети в дискретном времени (скалярная форма).
4. Напишите векторно-матричные уравнения, характеризующие динамику рекуррентной нейронной сети в дискретном времени.
5. В чем состоит задача распознавания образов? Какая информация должна быть представлена об объекте при решении задачи распознавания образов?
6. Какую задачу решает сеть Хемминга?
7. В какой форме представлены характеристики объектов при решении задачи распознавания образов с помощью сети Хемминга?
8. Каким способом заданы классы при решении задачи распознавания образов с помощью сети Хемминга?
9. Сформулируйте критерий оптимальности отнесения объекта к одному из классов при применении сети Хемминга.
10. Какое содержание имеют выходные значения нейронов рабочего слоя сети Хемминга?
11. Сколько нейронов содержит рабочий слой сети Хемминга? Каковы их синаптические коэффициенты?
12. Как связаны расстояние по Хеммингу между эталонным представителем и классифицируемым входным вектором и потенциал соответствующего нейрона сети Хемминга?

13. Какая активационная характеристика используется в сети Хемминга? Как она может быть реализована?
14. Что называется автоассоциативной памятью? Приведите пример.
15. Что называется аттрактором динамической системы? Объясните принцип применения динамических систем с множеством аттракторов для построения ассоциативной памяти.
16. Напишите уравнения динамики сети Хопфилда.
17. Какова активационная характеристика нейронов сети Хопфилда?
18. Чему равно начальное состояние нейронов сети Хопфилда?
19. Объясните различие синхронного и асинхронного режимов функционирования рекуррентной нейронной сети. Какой из режимов функционирования используется в сети Хопфилда?
20. Каким выражением определен энергетический функционал в процессе работы сети Хопфилда?
21. Почему время достижения сетью Хопфилда одного из аттракторов из произвольного начального состояния конечно?
22. Как рассчитывается матрица синаптических коэффициентов сети Хопфилда? Какими свойствами она обладает?
23. Как приближенно оценивается объем памяти сети Хопфилда?
24. На какие типы можно разделить множество аттракторов сети Хопфилда? Что такое «ложная память»?
25. Как применяется сеть Хопфилда для кластеризации данных? Как интерпретируются кластеры?
26. Как устанавливается класс принадлежности вектора признаков при использовании нескольких (по числу классов) сетей Хопфилда, «настроенных» на разные классы?
27. Как организован в лабораторной работе процесс оценивания размеров бассейнов аттракторов сети Хопфилда?
28. В чем состоит проблема кластеризации данных? Какие выборочные данные необходимы для решения этой задачи?
29. Где в соревновательном слое нейронов хранится информация о прототипах кластеров?
30. Дайте описание работы нейронов соревновательного слоя.
31. В чем состоит избирательность нейронов соревновательного слоя по отношению к данным обучающей выборки?
32. Как оценивается расстояние между выборочным примером и вектором синаптических коэффициентов нейрона?

33. Почему именно потенциалы нейронов соревновательного слоя используются для выявления нейрона-победителя?
34. Объясните правило самообучения нейронов соревновательного слоя.
35. Что называется «эпохой» самообучения соревновательного слоя нейронов?
36. Как влияет выбор параметра скорости самообучения на качество процесса настройки синаптических коэффициентов? Какие практические рекомендации можно дать для выбора этого параметра?
37. Что такое «соревновательная» активационная характеристика?
38. Какие негативные особенности можно ожидать при реализации самообучения соревновательного слоя нейронов?
39. Каким способом можно избежать появления «мертвых» нейронов соревновательного слоя?
40. По каким причинам может возникать укрупнение кластеров при обучении соревновательного слоя нейронов?
41. Как практически можно оценить качество обучения соревновательного слоя нейронов? Каким свойством обладает оптимально обученный соревновательный слой?
42. Какое основное качество отличает карту Кохонена от соревновательного слоя нейронов?
43. Приведите примеры оценки расстояния между нейронами в топографической карте. Какими в этих примерах являются линии точек, равноудаленных от фиксированного нейрона?
44. В чем состоит правило обучения нейронов карты Кохонена? Каково его отличие от правила обучения нейронов соревновательного слоя?
45. В чем состоит «раскраска» топографической карты отдельными признаками?
46. Что представляет собой одномерная карта Кохонена?
47. Какой вы можете привести пример практического применения карты Кохонена?
48. Напишите уравнения функционирования многослойной нейронной сети.
49. Объясните состав данных таблицы обучающей выборки, используемой для настройки параметров нейронной сети.
50. Какие параметры многослойной нейронной сети настраиваются в процессе ее обучения?
51. Какой критерий используется для организации обучения многослойной нейронной сети?
52. Чем характеризуется эффект «паралича» при обучении многослойной нейронной сети?
53. В чем состоит и как реализуется процесс инициализации при обучении нейронной сети?

54. В чем состоит и как проверяется эффект «генерализации данных» в нейронной сети?
55. Какие параметры режима обучения многослойной нейронной сети доступны пользователю для настройки?
56. В чем состоит процедура валидации многослойной нейронной сети?
57. В чем состоит процедура тестирования обученной многослойной нейронной сети?
58. Какие переменные вычисляются с помощью метода обратного распространения ошибки и как они используются в процессе обучения многослойной нейронной сети?
59. Чем отличаются пакетный и стохастический режимы обучения многослойной нейронной сети?
60. По какому правилу производится модификация значений параметров многослойной нейронной сети при ее обучении методом обратного распространения ошибки?
61. Объясните содержание задачи аппроксимации функции многих переменных на многослойной нейронной сети.
62. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче аппроксимации функции многих переменных?
63. Объясните содержание задачи прогноза временных рядов на многослойной нейронной сети.
64. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче прогноза временных рядов?
65. Объясните содержание задачи классификации данных на многослойной нейронной сети.
66. В чем состоит этап подготовки данных для обучения многослойной нейронной сети в задаче классификации данных?
67. Какой критерий используется при решении задачи классификации данных с помощью многослойной нейронной сети?
68. Какие правила останова процесса обучения МНС применяются в практических приложениях?
69. В чем состоит функциональная задача дополнительного блока принятия решения на выходе МНС при использовании ее в качестве классификатора данных?
70. Почему чрезмерное увеличение числа нейронов скрытых слоев в МНС при решении задачи аппроксимации может ухудшить точностные показатели сети? Какое свойство нейронной сети имеется в виду?
71. Почему на задачи классификации данных и прогноза временного ряда распространяются те же рекомендации по выбору архитектуры МНС, что и в задаче аппроксимации функции многих переменных?
72. Какие вы можете предложить нейросетевые решения задачи прогноза временного ряда одновременно на несколько последовательных временных тактов?

73. Предложите вариант построения нейросетевого классификатора при числе классов 4 с использованием многослойных нейросетевых классификаторов на 2 класса.

Критерии оценки:

Оценка «отлично» выставляется студенту, если он, в полном объеме знает состав, порядок установки и настройки системного и прикладного программного обеспечения, режимы его взаимодействия с операционной системой, умет настраивать операционную систему и выбирать отказоустойчивую конфигурацию компьютерных систем, оценивает все режимы функционирования сетей и систем передачи информации с помощью аппаратнопрограммных средств.

Оценка «хорошо» выставляется студенту, если он, знает состав, порядок установки и настройки системного и прикладного программного обеспечения, режимы его взаимодействия с операционной системой, умет с ошибками настраивать операционную систему и выбирать отказоустойчивую конфигурацию компьютерных систем, оценивает основные режимы функционирования сетей и систем передачи информации с помощью аппаратнопрограммных средств.

Оценка «удовлетворительно» выставляется студенту, если он, В основном знает состав, порядок установки и настройки системного и прикладного программного обеспечения, режимы его взаимодействия с операционной системой, умет настраивать операционную систему и выбирать отказоустойчивую конфигурацию компьютерных систем, оценивает минимальнодопустимые режимы функционирования сетей и систем передачи информации с помощью аппаратнопрограммных средств.

Оценка «неудовлетворительно» выставляется студенту, на низком уровне знает состав, порядок установки и настройки системного и прикладного программного обеспечения, режимы его взаимодействия с операционной системой, умет настраивать операционную систему и выбирать отказоустойчивую конфигурацию компьютерных систем, не оценивает минимальнодопустимые режимы функционирования сетей и систем передачи информации с помощью аппаратнопрограммных средств. Компетенция ОПК-2 не освоены.

Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным 55. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
---	--

Отличный	100
Хороший	80
Удовлетворительный	60
Неудовлетворительный	0

5.6. Методические рекомендации по подготовке к зачетам

Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка выполнения доклада и его презентации.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

6. Список литературы для выполнения СРС

Перечень основной литературы

1. Нейросетевые технологии обработки информации : учеб. пособие : Направление подготовки 010200.62 - Математика и компьютерные науки. Профиль подготовки - Вычислительная математика, информатика и компьютерные технологии. Бакалавриат / Аведьян Э. Д., Галушкин А. И., Червяков Н. И., Сахнюк П. А. ; Севю-Кав. федер. ун-т. МФТИ. - Ставрополь : СКФУ, 2014. - 283 с. : ил. - Гриф: Рек. УМО. - Библиогр.: с. 277-281
2. Барский, А.Б. Введение в нейронные сети Электронный ресурс : учебное пособие / А.Б. Барский. - Введение в нейронные сети, 2019-12-01. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 358 с. - Книга находится в базовой версии ЭБС IPRbooks.

Перечень дополнительной литературы

1. Барский, А.Б. Логические нейронные сети Электронный ресурс : учебное пособие / А.Б. Барский. - Логические нейронные сети, 2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 492 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-94774-646-4
2. Применение искусственных нейронных сетей и системы остаточных классов в криптографии / [авт. кол.: Червяков Н.И., Евдокимов А.А., и др.]. - Москва : ФИЗМАТЛИТ, 2012. - 280 с. : ил. - Прил.: с. 269-279. - Библиогр.: с. 269-279 (191 назв.) и в конце гл. - ISBN 978-5-9221-1386-1

Перечень учебно-методического обеспечения самостоятельной работы обучающихся по дисциплине

1. Методические указания по организации самостоятельной работы студентов по дисциплине «Искусственные нейронные сети». Ставрополь : СКФУ, 2025.
2. Методические указания к практическим занятиям по дисциплине «Искусственные нейронные сети». Ставрополь : СКФУ, 2025.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. www.aiportal.ru/ Портал искусственного интеллекта.
2. www.compres.ru. Сайт журнала «Компьютер пресс».
3. www.computer-museum.ru. Ежедельник PC-week RE и виртуальный компьютерный музей.
4. www.ibxt.ru. Новости вычислительной техники.
5. www.ibxt.ru. Новости вычислительной техники.
6. www.intuit.ru. Интернет-университет информационных технологий ИНТУИТ.
7. www.osp.ru. Издательство «Открытые системы».
8. www.radiotec.ru/catalog.php?cat=jr7 Журнал «Нейрокомпьютеры: разработка, применение».
- 9.