

Министерство науки и высшего образования Российской Федерации
Сибирский государственный университет науки и технологий
имени академика М. Ф. Решетнева

М. Г. Доррер

МОДЕЛИРОВАНИЕ НЕЙРОННЫХ СЕТЕЙ В СИСТЕМЕ MATLAB

*Лабораторный практикум для студентов бакалавриата
по направлениям подготовки*

09.03.01 «Информатика и вычислительная техника»,

09.03.04 «Программная инженерия»

очной и заочной форм обучения

Красноярск 2021

УДК 004.032.26(076.5)

ББК 32.813.5я73

Д69

Рецензенты:

доктор технических наук, профессор Е. В. Смирнова

(Сибирский федеральный университет);

доктор технических наук, профессор И. М. Горбаченко

(Сибирский государственный университет науки и технологий

имени академика М. Ф. Решетнева)

Доррер, М. Г.

Д69 Моделирование нейронных сетей в системе MatLab : лабораторный практикум / М. Г. Доррер ; СибГУ им. М. Ф. Решетнева. – Красноярск, 2021. – 98 с.

Дается краткое представление о работе в системе MatLab 2016, краткое теоретическое введение в курс искусственных нейронных сетей (ИНС). Рассматриваются вопросы подготовки данных для работы с ИНС, также вопросы проектирования структур ИНС, их обучения и использования для задач обработки данных. Пособие формирует навыки работы с известными типами нейронных сетей в системе MatLab 2016.

Лабораторный практикум подготовлен на кафедре «Информационно-управляющих систем» и предназначен для студентов направлений подготовки 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия» при изучении дисциплин «Нейрокомпьютерные системы» и «Нейросетевые технологии».

Также рекомендуется студентам, специализирующимся в области искусственного интеллекта и обработки больших данных (BigData).

УДК 004.032.26(076.5)

ББК 32.813.5я73

© СибГУ им. М. Ф. Решетнева, 2021

© Доррер М. Г., 2021

СОДЕРЖАНИЕ

Введение	5
<i>Лабораторная работа 1. Освоение базовых навыков работы с системой MatLab 2016</i>	<i>7</i>
1.1. Базовые сведения о системе MatLab	7
1.2. Интерфейс MatLab	8
1.3. Работа с векторами и матрицами в MatLab	13
1.4. Программирование в среде MatLab	20
Порядок выполнения работы	26
Индивидуальные задания	27
Содержание отчета	28
Контрольные вопросы и задания	28
<i>Лабораторная работа 2. Классификация с помощью персептрона</i>	<i>29</i>
2.1. Нейронные сети: основные положения	29
2.2. Однослойный персептрон	33
2.3. Классификации однослойным персептроном на 2 класса	37
2.4. Классификация однослойным персептроном на 4 класса	42
Порядок выполнения работы	46
Индивидуальные задания	47
Содержание отчета	47
Контрольные вопросы и задания	48
<i>Лабораторная работа 3. Аппроксимация функции при помощи ИНС</i>	<i>49</i>
3.1. Постановка задачи аппроксимации	49
3.2. Реализация многослойной сети в MatLab	54
Порядок выполнения работы	58
Индивидуальные задания	59
Содержание отчета	59
Контрольные вопросы и задания	60
<i>Лабораторная работа 4. Классификация с помощью сетей Кохонена</i>	<i>61</i>
4.1. Классификация без учителя при помощи ИНС	61
4.2. Пример решения задачи классификации сетью Кохонена	67

Порядок выполнения работы	71
Индивидуальные задания	72
Содержание отчета	73
Контрольные вопросы и задания	73
 <i>Лабораторная работа 5. Нейросетевое прогнозирование</i>	
временных рядов	74
5.1. Понятие прогнозирования временных рядов	74
5.2. Определение структуры нейронной сети для прогноза ВР	75
5.3. Подготовка обучающей выборки для прогнозирования ВР	77
5.4. Построение программы в среде MatLab, использующей нейронные сети для прогнозирования временных рядов	78
Порядок выполнения работы	80
Индивидуальные задания	81
Содержание отчета	81
Контрольные вопросы и задания	82
 <i>Лабораторная работа 6. Сети Хопфилда</i>	
6.1. Теоретические аспекты сетей Хопфилда	83
6.2. Реализация ИНС Хопфилда	85
Порядок выполнения работы	87
Варианты индивидуальных заданий	87
Содержание отчета	88
Контрольные вопросы и задания	88
 <i>Лабораторная работа 7. Нейросетевое распознавание</i>	
изображений	89
7.1. Задача распознавания образов	89
7.2. Решение задачи распознавания символов	90
Порядок выполнения работы	94
Варианты индивидуальных заданий	95
Содержание отчета	95
Контрольные вопросы и задания	95
 Заключение	96
 Библиографический список	97

ВВЕДЕНИЕ

Искусственные нейронные сети (ИНС) – область теории и практики искусственного интеллекта, в которой обработка сигналов ведется по формулам и алгоритмам, моделирующим явления, происходящие в нейронах нервной системы живых существ.

Первым важнейшим преимуществом нейронных сетей, повышающим эффективность использования вычислительных ресурсов и обеспечивающим ей преимущества над традиционными архитектурами в целом ряде вычислительных задач, является параллельная обработка информации всеми звеньями сети (так называемый «мелкозернистый параллелизм»). Громадное количество нейронов и еще большее количество межнейронных связей (синапсов) в сочетании с их параллельной работой позволяет значительно ускорить процесс обработки информации. Такая вычислительная архитектура зачастую позволяет решать задачи обработки сигналов (в том числе, таких сложных, как видео и аудио) в реальном времени. Большое количество синапсов придает ИНС свойство «стереоскопичности» – устойчивости к ошибкам в некоторых синапсах и даже к их разрыву. Функции поврежденных синапсов берут на себя оставшиеся, в результате чего оставшаяся система сохраняет способность к решению задач.

Второе важное свойство ИНС – способность (ранее присущая лишь живым организмам) к обучению, в том числе и дообучению по вновь поступившей информации, а также к обобщению накопленных знаний. Нейронная сеть, обученная на ограниченном наборе данных, способна производить обобщение полученной информации и с приемлемым качеством решать задачи на основании данных, ранее не предъявлявшихся ей в процессе обучения. Это придает нейронным сетям очень важное для целого ряда приложений свойство возможности решения задач в неполных и плохо формализованных данных, а также позволяет решать обратную задачу восстановления неполных данных.

С точки зрения аппаратной реализации характерной особенностью ИНС является возможность ее реализации с применением технологии сверхбольшой степени интеграции. Вычислительные элементы сети однотипны, их повторяемость огромна. Используя это свойство можно как эффективно реализовывать модели нейронных сетей на существующих высокопараллельных архитектурах (параллельные и кластерные системы, ядер CUDA – скалярных вычислительных блоков в видео-чипах NVidia), так и создавать аппаратные нейросети – универсальные процессоры с однородной структурой, способные в высокопараллельном режиме перерабатывать разнообразную информацию.

В практикуме кратко представлены теоретические сведения об основных парадигмах и архитектурах ИНС, а также представлены примеры их реализации и решения типовых задач в компьютерной среде математического моделирования MatLab на примере версии 2016 г.

В результате изучения дисциплины студенты должны:

знать основные парадигмы и архитектуры ИНС и области их применения; основные способы, приемы и алгоритмы обучения ИНС;

уметь моделировать ИНС средствами нейропакета MatLab NNTool; оценивать качество обучения различных моделей ИНС на имеющихся экспериментальных данных;

владеть навыками решения задач аппроксимации, прогнозирования, кластеризации и классификации данных, распознавания образов при помощи ИНС.

В результате освоения дисциплины «Моделирование нейронных сетей в системе MatLab» формируются следующие профессиональные компетенции (ПК):

1) 09.03.01 «Информатика и вычислительная техника»:

способность обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности (ПК-3);

2) 09.03.04 «Программная инженерия»:

способность к формализации в своей предметной области с учетом ограничений используемых методов исследования (ПК-12).

Курс «Моделирование нейронных сетей в системе Matlab» связан с другими дисциплинами следующим образом.

Курс занимает место в ряду дисциплин, посвященных освоению методов и средств анализа данных и искусственного интеллекта. Курс опирается на следующие дисциплины: «Математический анализ», «Вероятностные методы в программной инженерии», «Математические основы искусственного интеллекта», «Дополнительные главы математики», «Основы математического программирования», «Численные методы».

Данный курс призван выработать у студентов целостное представление о механизмах и инструментах искусственных нейронных сетей. Разработку ИНС следует рассматривать как научную и инженерную задачу, а саму обученную на решение определенного класса задач ИНС с наборами обучающих и тестовых данных – как технологический объект.

Начнем знакомиться с данным курсом с основных понятий.

Лабораторная работа 1
**ОСВОЕНИЕ БАЗОВЫХ НАВЫКОВ РАБОТЫ
С СИСТЕМОЙ MATLAB 2016**

Цель работы: освоение обучающимися интерфейса и основных функциональных возможностей системы MatLab. Приобретение навыков, необходимых для начала работы с нейросетевым расширением.

1.1. Базовые сведения о системе MatLab

Система MatLab представляет собой высокоуровневый язык математического программирования, интегрированный в интерактивную среду разработки. Главное назначение системы – выполнение численных расчетов и визуализации результатов. С помощью системы MatLab можно выполнять анализ данные, используя встроенные в систему специализированные расширения разрабатывать собственные алгоритмы, создавать математические модели процессов и систем, аналитические и расчетные приложения.

Система нашла широкое применение в целом ряде областей знания:

- обработка сигналов и связь;
- искусственный интеллект;
- обработка изображений и видео;
- системы управления;
- автоматизация тестирования и измерений;
- имитационное моделирование процессов;
- финансовое прогнозирование;
- аналитические расчеты в биологии и т. п.

Вычислительный «движок» системы обеспечивает простую и удобную работу с матрицами реальных, комплексных и аналитических типов данных, со структурами данных и таблицами поиска.

MatLab содержит встроенные функции линейной алгебры, быстрых преобразований Фурье (БПФ), функции для работы с многочленами, функции математической статистики, численного (а с применением расширений – и аналитического) решения дифференциальных уравнений.


Работа в среде MatLab производится в следующих двух режимах.

1. Интерактивный режим. Вычисления осуществляются непосредственно после набора очередной команды MatLab, полученные в результате расчетов присваиваются некоторым переменным.

2. Программный режим. Работа ведется путем вызова программы, написанной на языке MatLab (файл с расширением «mat»), содержащей полный набор команд, обеспечивающих ввод данных, выполнение расчетов и вывод результатов на экран.

1.2. Интерфейс MatLab

Рабочая среда MatLab

Запуск системы производится двойным кликом по иконке . Откроется интерфейсное окно рабочей среды (рис. 1.1). В этом окне представлены обычные средства управления размерами, скрытия и закрытия. В окне командного режима показано окно About MATLAB, которое выводится одноименной командой в позиции справочного меню (Help) и позволяет уточнить версию системы.

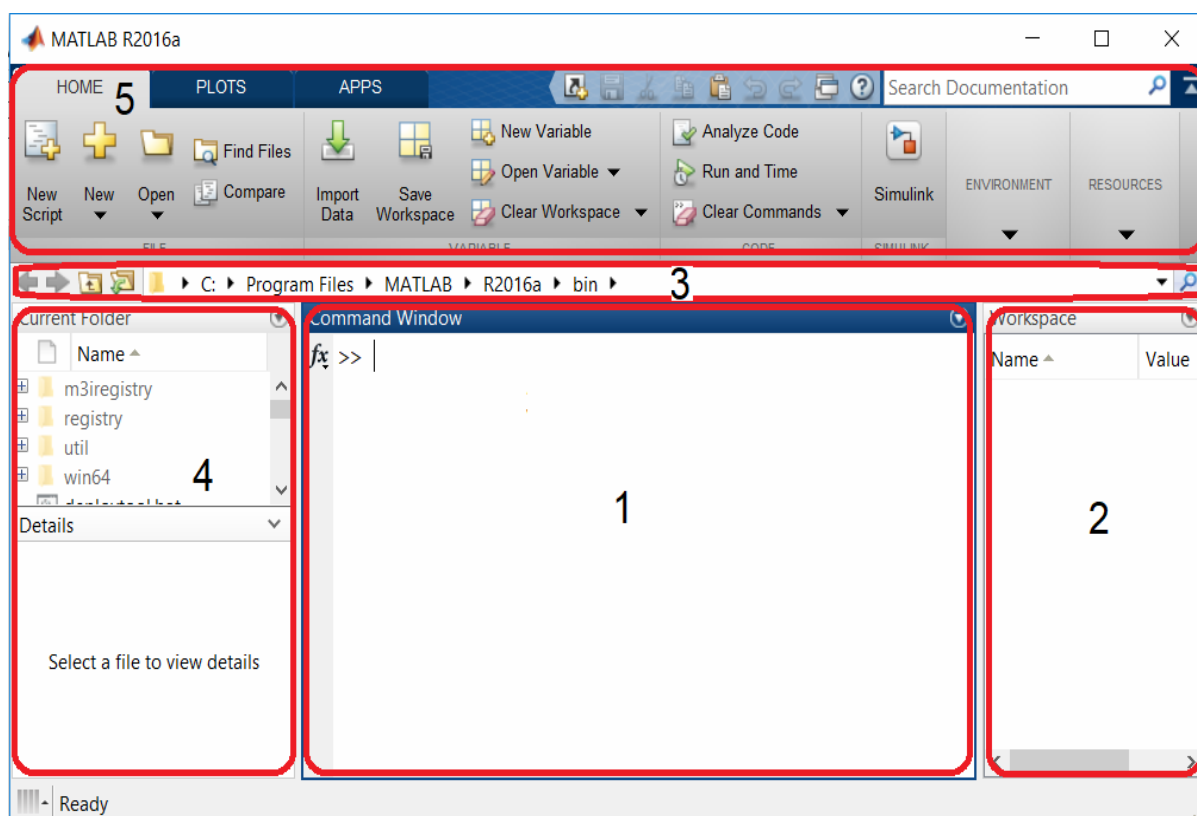


Рис. 1.1. Основные элементы пользовательского интерфейса MatLab

Система готова к проведению вычислений в командном режиме. Полезная информация. В начале запуска системы автоматически выполняется команда **matlabrc**, которая исполняет загрузочный файл **matlabrc.m** и файл **startup.m** (если есть). Эти файлы текстового

формата, содержащие программы на языке MatLab, выполняют начальную настройку терминала системы и задают ее начальные параметры.

Рассмотрим основные элементы графического интерфейса MatLab, отмеченные на рис. 1.1, и кратко опишем их назначение.

В рабочую среду системы MatLab входят следующие элементы:

1. Командное окно, предназначенное для интерактивной работы в MatLab.
2. Рабочая область, в которой отображаются текущие переменные.
3. Рабочий каталог, где содержатся пользовательские скрипты и функции.
4. Окно с вкладками *Command History* и *Current Folder*, предназначенное для просмотра и повторного вызова ранее введенных команд, а также для установки текущего каталога.
5. Окно с вкладками *Launch Pad* и *Workspace*, из которого можно получить доступ к различным модулям *ToolBox* и к содержимому рабочей среды.

Настройка состава окон производится с использованием пунктов меню *View: Command Window, Command History, Current Directory, Workspace, Launch Pad*.

Ввод команд производится в командном окне после символа «>>».

Набор любой команды или выражения в MatLab должен заканчиваться нажатием клавиши «**Enter**». После этого ядро MatLab выполняет команду или вычисляет введенное выражение.

Простейшие вычисления

Выполним ввод простейшего выражения – $2*2$ (рис. 1.2). Введем его в командной строке и нажмем **Enter**. Командное окно MatLab после этого выглядит следующим образом:

Видно, что систем MatLab вычислила произведение $2*2$, результат занесла в предопределенную переменную *ans*, а ее значение, равное 4, вывела в командное окно. Ниже ответа расположена командная строка с мигающим курсором и «>>» – приглашением к вводу новых команд.

Результаты, хранящиеся в переменной *ans*, можно использовать в последующих расчетах. При вводе выражения $ans/4$ и **Enter**, система выдаст следующий результат:

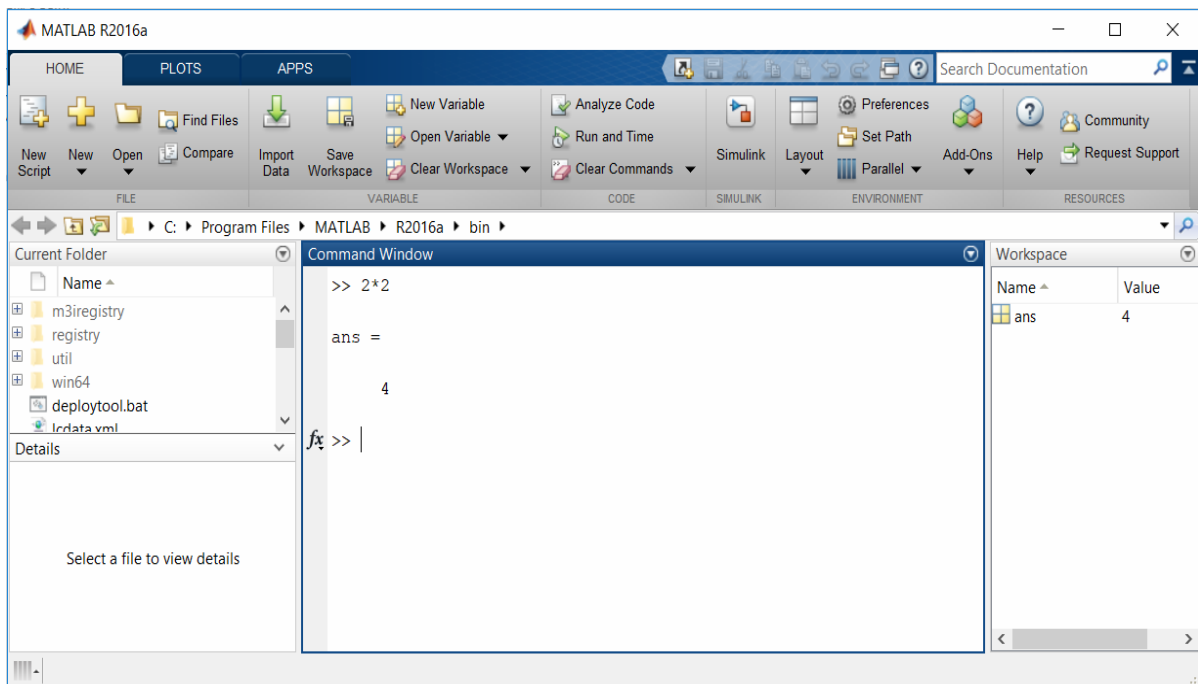


Рис. 1.2. Простейшее выражение в MatLab

Отклик команд в MatLab

В результате выполнения каждой команды системой в командное окно выводится отклик команды, ее «эхо». На рис. 1.3 это ответ *ans* = 1. Отклики команд загромождают командное окно и затрудняют отслеживание результатов. Для того чтобы от этого избавиться, отклик команд можно отключить, завершив команду символом «;» («точка с запятой») (рис. 1.4).



Рис. 1.3. Использование для расчета переменной *ans*

Видно, что за командой «ans*4;» отклика не последовало.




Рис. 1.4. Подавление отклика команд в MatLab
с помощью «;»


Сохранение параметров рабочей среды MatLab.

Работа с MAT-файлами

Результаты работы MatLab сохраняются в файле с расширением *mat*.

Рабочую среду – состав переменных, их значения с типами данных – эффективнее всего сохранить при помощи команды **Save**

Workspace  из панели инструментов **Home**. Для этого используется диалоговое окно **Save Workspace**, в котором можно выбрать или ввести путь к расположению файла и его имя. По умолчанию предлагается сохранить файл в подкаталоге «**Program Files\MATLAB\R2016a\bin**» основного каталога MatLab.

Для восстановления значений переменных в следующих сеансах работы используется команда **Open**  панели инструментов **Home**, позволяющая открыть сохраненный файл.

Журнал команд

В системе MatLab предусмотрена возможность ведения журнала работы – записи исполняемых команд и результатов расчета в текстовый файл («дневник»), к изучению которого можно вернуться позже. Ведение журнала операций запускается командой **diary**. Аргументом данной команды является имя файла, предназначенного для сохранения журнала. После вызова команды **diary** все команды и результаты их исполнения система записывает в указанный в аргументе файл. Пример работы с дневником MatLab приведен на рис. 1.5.

Следует помнить, что по умолчанию система попытается сохранить файл дневника в рабочую папку MatLab. А поскольку она расположена в каталоге *Program Files*, то вполне возможно, что доступа на запись в эту папку не будет. Необходимо поменять рабочую папку в окне *Current Folder*.

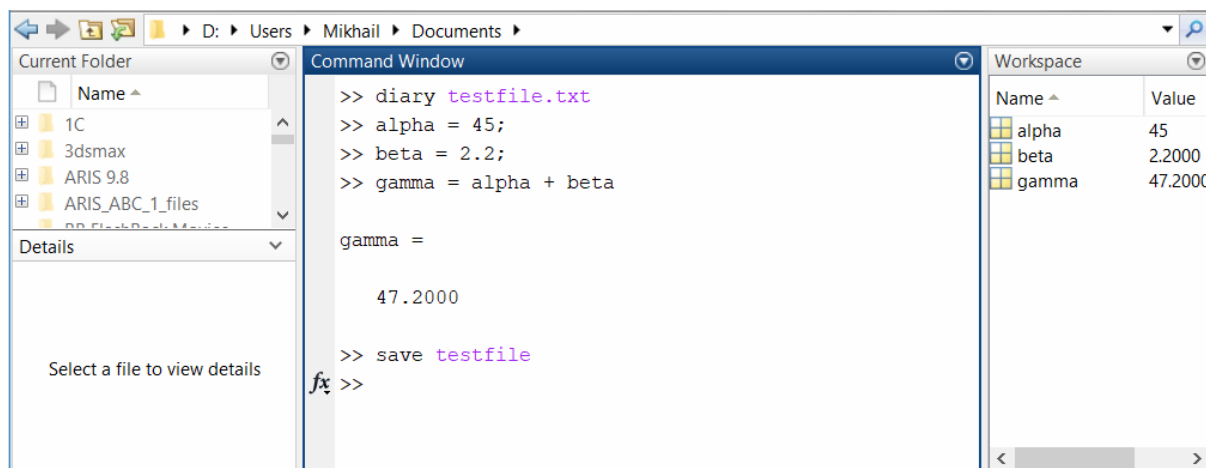


Рис. м5. Запись дневника операций MatLab

В примере, приведенном на рис. 1.5, система производит следующие действия:

1. Открывает журнал в файле «testfile.txt».
2. Производит вычисления.
3. Сохраняет все переменные в MAT файле «testfile.mat».
4. Сохраняет журнал в файле «testfile.txt» в выбранном подкаталоге.

Просмотрев содержимое файла «testfile.txt» в текстовом редакторе можно убедиться, что в нем сохранился следующий текст (рис. 1.6).

```
alpha = 45;
beta = 2.2;
gamma = alpha + beta

gamma =

47.2000

save testfile
```

Рис. 1.6. Содержимое дневника операций MatLab

Справочная система MatLab

Окно справки MatLab появляется после нажатием кнопки *Help*



на панели инструментов. Страница помощи может быть так же вызвана путем набора команды *helpwin* в командной строке. Окно справки по разделам вызывается путем набора команды *helpwin topic*.

Дополнительные справочные материалы можно найти на сайте Math Works, компании-разработчика системы MatLab (URL: <https://www.mathworks.com/>).

Большое количество русскоязычных справочных материалов представлено на сайте «Экспонента» (URL: <https://exponenta.ru/>).

1.3. Работа с векторами и матрицами в MatLab

Переменные скалярного, векторного и матричного типа

Численные значения в MatLab могут быть представлены в формах скаляров, векторов и матриц. При вводе скалярного значения достаточно приписать его значение какой-то переменной, например так, как это показано на рис. 1.7. Имена переменных в MatLab зависят от регистра буквы, то есть «A» и «a» – это разные переменные.



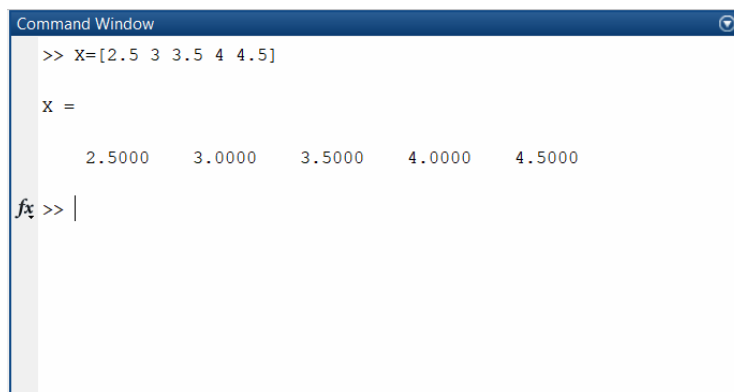
Рис. 1.7. Ввод скалярного значения

Ввод значений массивов (векторов или матриц) выполняется в квадратных скобках. На рис. 1.8 показан ввод вектора-строки размерностью 1×5 . Отдельные значения в строке разделяются запятыми или пробелами.

В векторе-столбце компоненты разделяют символом «;» (точка с запятой). На рис. 9 показан ввод вектора-столбца размерностью 4×1 .

При вводе матриц с клавиатуры они записываются как вектор-столбец, состоящий из векторов-строк или как вектор-строка, состоя-

щий из векторов-столбцов. На рис. 1.10 матрица 3×3 вводится этими двумя способами.



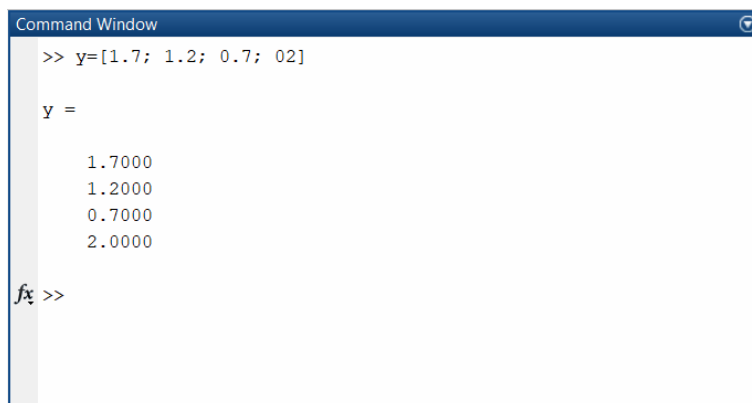
```
Command Window
>> x=[2.5 3 3.5 4 4.5]

x =

    2.5000    3.0000    3.5000    4.0000    4.5000

fx >> |
```

Рис. 1.8. Ввод вектора-строки



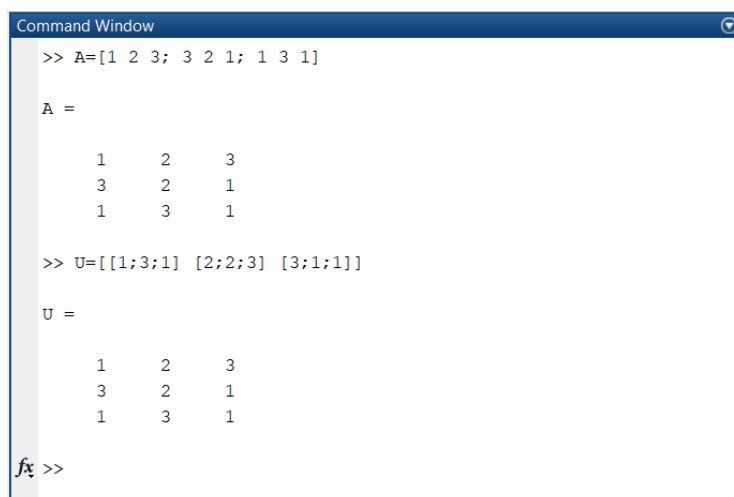
```
Command Window
>> y=[1.7; 1.2; 0.7; 0.2]

y =

    1.7000
    1.2000
    0.7000
    2.0000

fx >>
```

Рис. 1.9. Ввод вектора-столбца



```
Command Window
>> A=[1 2 3; 3 2 1; 1 3 1]

A =

     1     2     3
     3     2     1
     1     3     1

>> U=[[1;3;1] [2;2;3] [3;1;1]]

U =

     1     2     3
     3     2     1
     1     3     1

fx >>
```

Рис. 1.10. Ввод матрицы 3×3

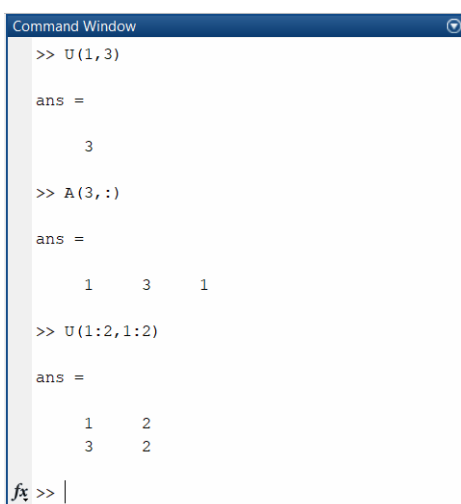
Индексация элементов векторов и матриц

Обращение к значению элемента матрицы в MatLab производится путем указания индекса элемента. Следует помнить, что работа может вестись с массивами любой размерности, например, для двумерного массива (матрицы) он осуществляется при помощи номеров строки и столбца, заключенных в круглые скобки. Так, для матрицы U с рис. 10 команда $U(1,3)$ выдаст значение ячейки, расположенной в первой строке, в ее третьем столбце.

Выделение подмножества матрицы с меньшей размерностью например, для двумерной – строки или столбца – производится путем замены одного из индексов на символ «:» (двоеточие). Запись $A(3,:)$ адресует к 3-й строке матрицы A с рис. 1.10.

Блок в матрице можно выделить при помощи двоеточия. Например, при помощи записи $U(1:2,1:2)$ будет выделен из матрицы U с рис. 1.11 левый верхний блок размером 2×2 элемента.

Просмотр списка переменных рабочей среды производится вводом в командную строку команды **whos** (рис. 1.12).



```
Command Window
>> U(1,3)

ans =

     3

>> A(3,:)

ans =

     1     3     1

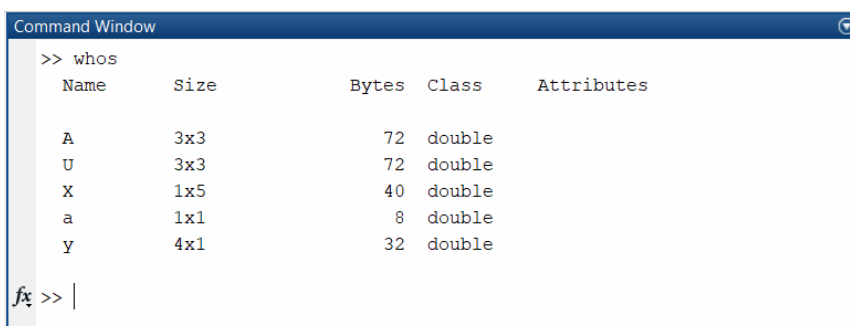
>> U(1:2,1:2)

ans =

     1     2
     3     2

fx >> |
```

Рис. 1.11. Адресация элементов матриц



```
Command Window
>> whos

Name      Size      Bytes  Class  Attributes

A         3x3         72  double

U         3x3         72  double

X         1x5         40  double

a         1x1          8  double

y         4x1         32  double

fx >> |
```

Рис. 1.12. Просмотр переменных среды

В результате действий на рис. 1.7–1.11 в рабочей среде сформировалось 5 переменных – 2 матрицы (A , U), два вектор (X , y) и один скаляр (a).

Выполнение матричных операций

Рассмотрим операции сложения и умножения матриц и векторов в MatLab. Пусть даны два вектора:

$a = [6 \ 5 \ 4 \ 3 \ 2]$; – вектор-строка

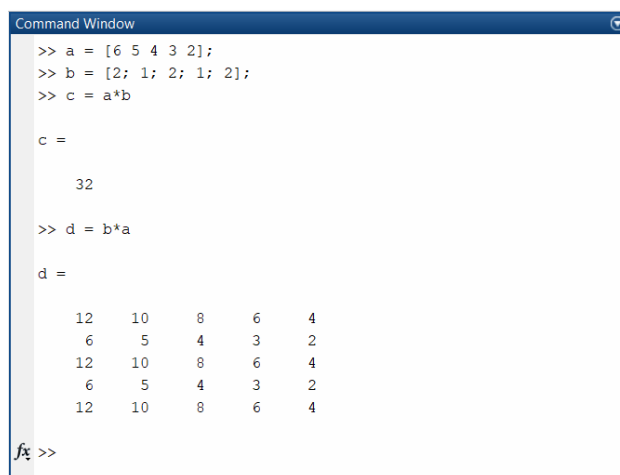
$b = [2; 1; 2; 1; 2]$; – вектор-столбец

Операция вычисления произведения этих двух векторов в зависимости от порядка вычисления даст следующий результат:

$c = a*b$; – результат c представляет собой скалярное значение;

$d = b*a$; – результат d представляет собой матрицу размерностью 5×5

Результаты этих вычислений приведены на рис. 1.13.



```
Command Window
>> a = [6 5 4 3 2];
>> b = [2; 1; 2; 1; 2];
>> c = a*b

c =

    32

>> d = b*a

d =

    12    10     8     6     4
     6     5     4     3     2
    12    10     8     6     4
     6     5     4     3     2
    12    10     8     6     4

fx >>
```

Рис. 1.13. Произведение векторов

Расчет суммы и разности двух векторов записывается следующим образом:

$v1 = [1 \ 2 \ 3 \ 4 \ 5]$;

$v2 = [5 \ 4 \ 3 \ 2 \ 1]$;

$x = v1+v2$;

$y = v2-v1$.

Результаты этих вычислений приведены на рис. 1.14.

При использовании матричных операций следует помнить, что для сложения или вычитания матрицы должны быть одного размера, а при перемножении число столбцов первой матрицы обязано равняться числу строк второй матрицы. В противном случае MatLab выдаст сообщение об ошибке, как и для любой недопустимой арифметической операции.

Операции умножения и сложения между матрицами выполняются аналогично (команда `ones(3)` создает матрицу размерности 3×3 , заполненную единицами).

$A = [3 \ 2 \ 1; 7 \ 5 \ 6; 9 \ 6 \ 8];$

$B = \text{ones}(3);$

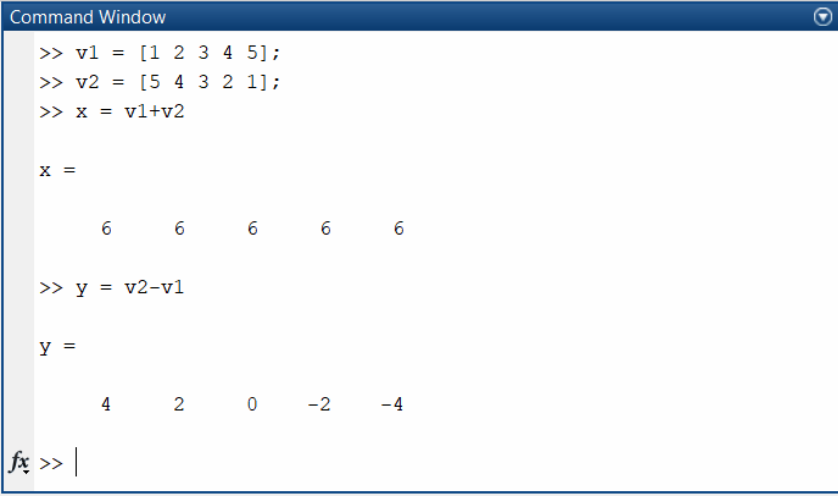
$C = A + B;$ – сложение двух матриц одинакового размера

$D = A + 5;$ – сложение матрицы со скаляром (число добавляется к каждому элементу)

$E = A * B;$ – умножение матрицы A на B

$F = B * A;$ – умножение матрицы B на A

$G = 3 * A;$ – умножение матрицы на скаляр



```

Command Window
>> v1 = [1 2 3 4 5];
>> v2 = [5 4 3 2 1];
>> x = v1+v2

x =

     6     6     6     6     6

>> y = v2-v1

y =

     4     2     0    -2    -4

fx >> |

```

Рис. 1.14. Сложение и вычитание векторов

В MatLab в виде отдельных команд заданы операции вычисления обратной матрицы, а также транспонирования матриц и векторов. Они записываются следующим образом (рис. 1.15).

```

a = [1 1 1]; % вектор-строка
b = a'; % вектор-столбец, образованный
% транспонированием вектора-строки a.
A = [1 2 3; 4 5 6; 7 8 9]; % матрица 3x3 элемента
B = a*A; % B = [12 15 18] – вектор-строка
C = A*b; % C = [6; 15; 24] – вектор-столбец
D = a*A*a'; % D = 45 – число, сумма элементов матрицы A
E = A'; % E – транспонированная матрица A
F = inv(A); % F – обратная матрица A
G = A^-1; % G – обратная матрица A

```

Рис. 1.15. Операции транспонирования и обращения матриц

Из приведенного примера видно, что операция транспонирования матриц и векторов обозначается символом «'» (апостроф), который ставится после имени вектора или матрицы. Вычисление обратной матрицы можно делать следующими тремя способами:

- путем вызова функции *inv()*;
- возводя матрицу в степень -1 ;
- деля единичную матрицу на заданную (операция, аналогичная возведению в -1 степень).

Результат во всех трех случаях будет одинаковым, а разные способы вычисления сделаны для удобства использования при реализации различных формул и алгоритмов.

Функция *pinv* позволяет вычислить псевдообратную матрицу.

Кроме того, в MatLab реализованы операции поэлементного умножения, деления и возведения в степень элементов многомерных массивов. Выполняются эти операторы при помощи команд:

- *.** – поэлементное умножение;
- *./* и *.* – поэлементные деления;
- *.^* – поэлементное возведение в степень.

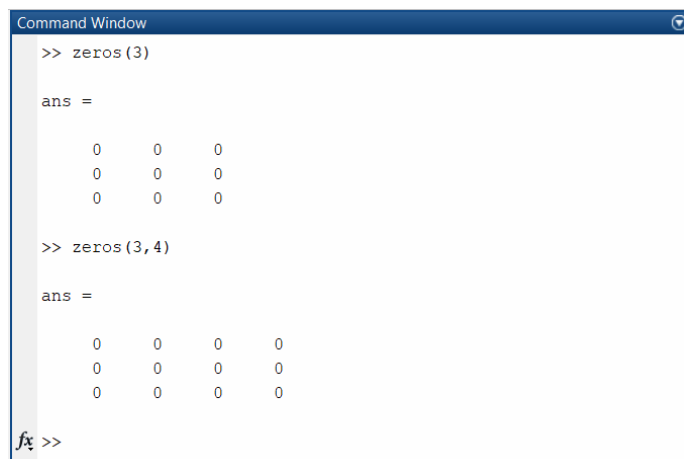
Работа с матрицами специального вида

Формирование прямоугольных матриц, заполненных нулями, производится при помощи функции *zeros*. Варианты ее применения показаны на рис. 1.16 – при указании одного индекса формируется нулевая квадратная матрица заданной размерности, а при указании 2-х – прямоугольная нулевая матрица заданной размерности.

Различные виды диагональных матриц можно получить при помощи функций *eye* и *diag*. Единичная матрица создается при помощи функции *eye*, в скобках указывается размерность требуемой матрицы, можно указывать 1 или 2 индекса аналогично функции *zeros*. Функция *diag* позволяет, в зависимости от аргумента, либо, получив вектор, сформировать матрицу с этим вектором на главной диагонали, либо, получив на вход матрицу, сформировать вектор из элементов ее главной диагонали. Примеры работы с диагональными матрицами приведены на рис. 1.17.

Генерация матриц со случайными элементами в MatLab выполняется при помощи функции *rand*. Функция *rand* генерирует матрицу из псевдослучайных элементов, равномерно распределенных между 0 и 1. При необходимости использовать нормальный закон распределения используется функция *randn*. Генерируемые псевдослучайные числа распределены по нормальному закону с нулевым средним и единичной дисперсией. В скобках указывается размерность требу-

мой матрицы, можно указывать 1 или 2 индекса. Примеры работы с матрицами псевдослучайных чисел приведены на рис. 1.18.



```
Command Window
>> zeros(3)

ans =

     0     0     0
     0     0     0
     0     0     0

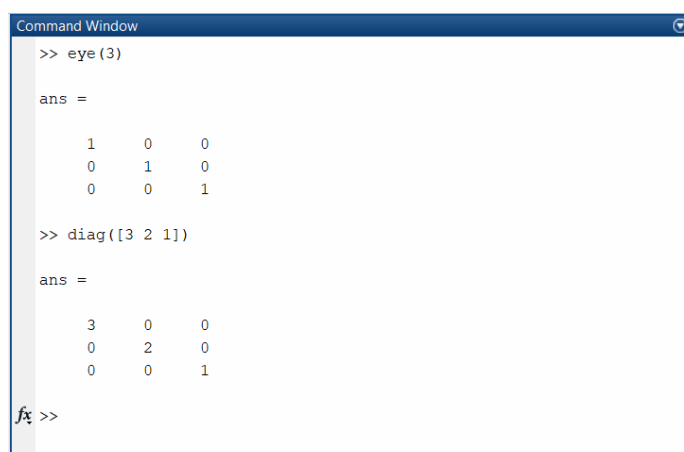
>> zeros(3,4)

ans =

     0     0     0     0
     0     0     0     0
     0     0     0     0

fx >>
```

Рис. 1.16. Использование функции zeros



```
Command Window
>> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1

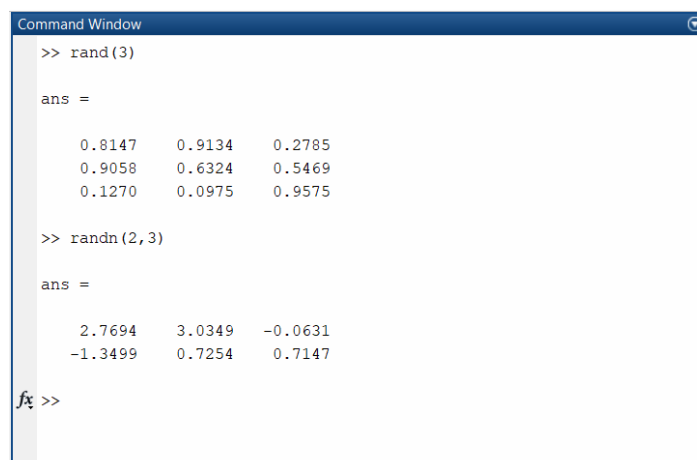
>> diag([3 2 1])

ans =

     3     0     0
     0     2     0
     0     0     1

fx >>
```

Рис. 1.17. Формирование диагональных матриц



```
Command Window
>> rand(3)

ans =

     0.8147     0.9134     0.2785
     0.9058     0.6324     0.5469
     0.1270     0.0975     0.9575

>> randn(2,3)

ans =

     2.7694     3.0349    -0.0631
    -1.3499     0.7254     0.7147



fx >>
```

Рис. 1.18. Построение матриц псевдослучайных чисел

1.4. Программирование в среде MatLab

Работа с М-файлами

Использование М-файлов повышает удобство и оперативность выполнения групп команд MatLab. Работа с М-файлами ведется в редакторе М-файлов. С его помощью можно создавать выполняемые вычислительные программы (*Script M-Files*), а также собственные функции (*Function M-Files*), которые можно вызывать как из командного окна, так и рекурсивно – из программ и других функций.

На вкладке Home нажмите пункт **New**  и подпункт **Script** .

Новый файл открывается в окне редактора М-файлов, на рис. 1.19, оно выделено рамкой.

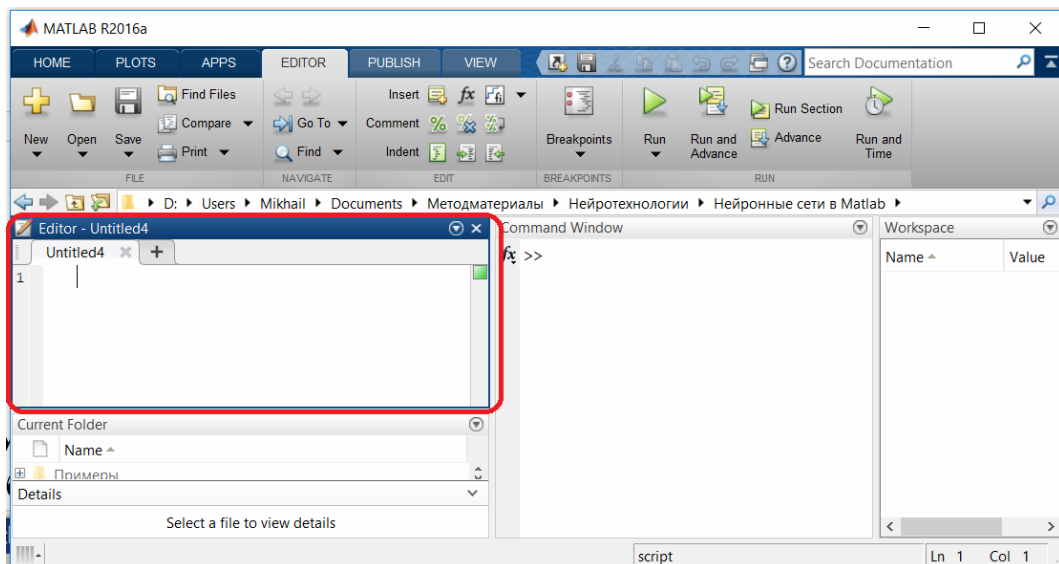


Рис. 1.19. Редактор М-файлов

Работа с М-файлами-программами

Введем в окне редактора простейшую программу (рис. 1.20), создающую две матрицы и суммирующую их.

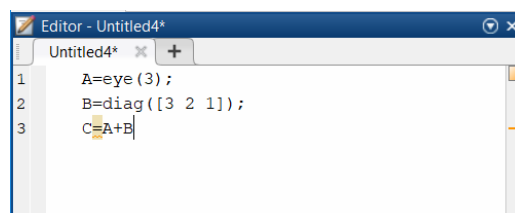

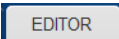



Рис. 1.20. Простейшая программа в MatLab

Сохраним файл с программы именем `testprog.m` в выбранном каталоге, выбрав пункт **Save**  вкладки **Editor** . Следует помнить, что MatLab при первом сохранении файла предлагает выбрать для него рабочую папку (рис. 1.21).

Ранее созданный М-файл открывается при помощи пункта **Open**  вкладки **Editor**.

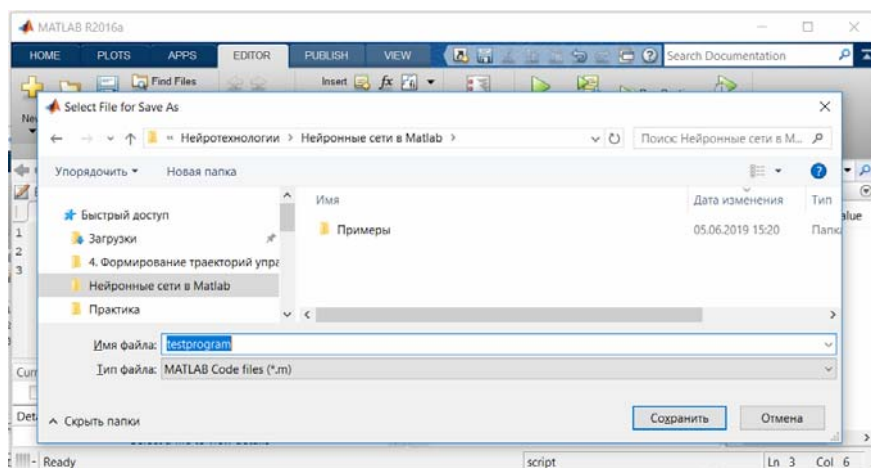



Рис. 1.21. Сохранение программы

Для запуска на выполнение программы, содержащихся в файле, следует выбрать пункт **Run**  на этой же вкладке. В командном окне (рис. 1.22) появится результат выполнения программы, а в окне рабочего пространства – вновь созданные переменные среды (в нашем случае – матрицы A , B и C).

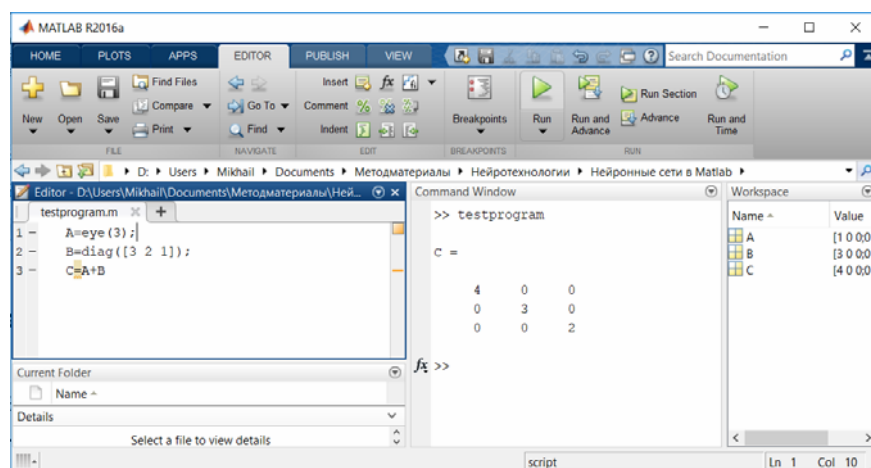


Рис. 1.22. Программа и результат ее выполнения

В случае, если в программе содержится ошибка и MatLab не может выполнить команду, то программа выполняется до строки, содержащей ошибку, а затем она останавливается, а в командное окно выводится сообщение об ошибке.

В редакторе М-файлов можно выполнить часть команд программы. Для этого необходимо, удерживая левую кнопку мыши, выделить часть команда и выполнить, нажав клавишу **F9**. На рис. 1.23 видно, что в результате выполнения 2 команд из 4, содержащихся в программе, в командное окно вывелась только матрица **C**, а в рабочем пространстве создались только две переменные – **B** и **C**.

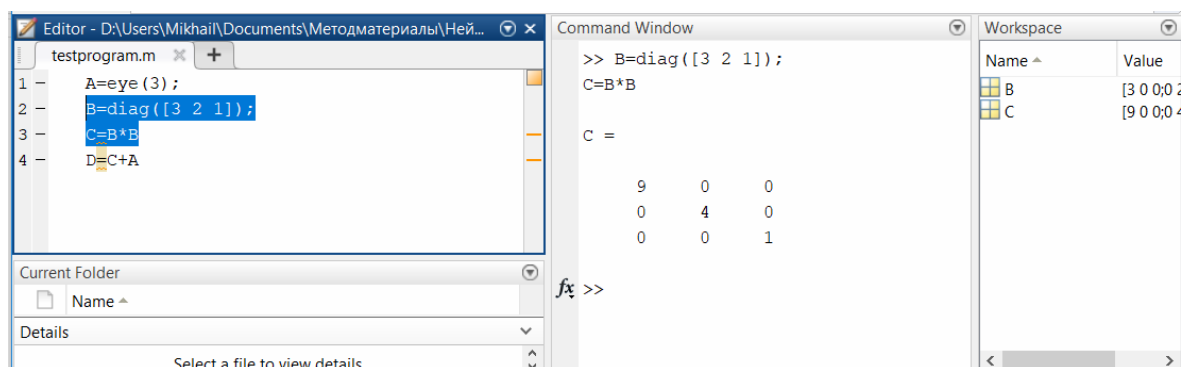


Рис. 1.23. Выполнение части программы в MatLab

При разработке программ в MatLab полезной функцией является возможность дополнить исполняемый текст комментариями, которые при выполнении пропускаются. Закомментированный текст начинается со знака процента. Для удобства работы с программой редактор автоматически выделяет закомментированный текст зеленым цветом. Пример комментариев – на рис. 1.24. Как видно, закомментированный текст может располагаться как в отдельной строке, так и следовать за выражениями MatLab.

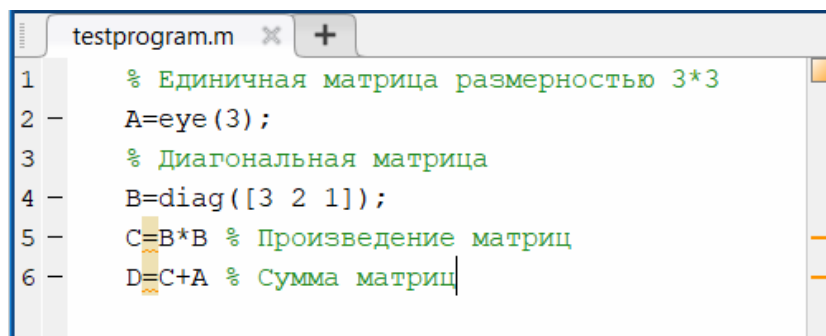


Рис. 1.24. Использование комментариев в MatLab

Работа с М-файлами-функциями

Файл-программа представляет собой набор команд MatLab, у нее отсутствуют входные и выходные аргументы. При разработке собственных расчетных и аналитических приложений в MatLab существенно полезной возможностью является использования файл-функций (*Function M-Files*). При этом один файл может содержать описание только одной функции, имя файла должно совпадать с именем функции, которая реализована в этом файле. Функции, как и в других языках программирования, производят требуемые манипуляции с входными аргументами и передают результаты в выходные аргументы.

Файл-функция (процедура) должна начинаться со строки заголовка:

function [<CP>] = <имя процедуры>(<CB>)

Если список результатов (CP) содержит только один объект, то М-файл представляет собой обычную функцию одной или нескольких переменных, которые перечисляются в списке входов (CB).

В этом случае первая строка должна иметь вид:

function <имя переменной> = <имя процедуры>(<CB>)

Если же в результате выполнения М-файла должны быть определены несколько объектов (скаляров, векторов, матриц), то такая файл-функция представляет собой процедуру (подпрограмму). Общий вид первой строки в этом случае становится таким:

function [y1, y2, ... , yn] = <имя процедуры>(<CB>)

В этом случае перечень список результатов y1, y2, ..., yn должен быть представлен как вектор-строка с соответствующими элементами, каждый из которых может быть матрицей.

В качестве примера рассмотрим составление М-файла для функции $y = a(\sin(x^2) + \cos(x^2))$ (рис. 1.25).

```
function y = TestFunc(x,a)
% Процедура, которая вычисляет значение функции
% y = a*(sin(x^2)+cos(x^2))
% Обращение y = TestFunc(x,a)
y = a*(sin(x^2)+cos(x^2));
```

Рис. 1.25. Пример функции двух аргументов

После ввода текста в окне редактора необходимо сохранить его в файле под именем **TestFunc.m**. Теперь если ввести в командном окне команду **y=TestFunc(2,25)**, то ядро MatLab произведет расчет так, как это показано на рис. 1.26.

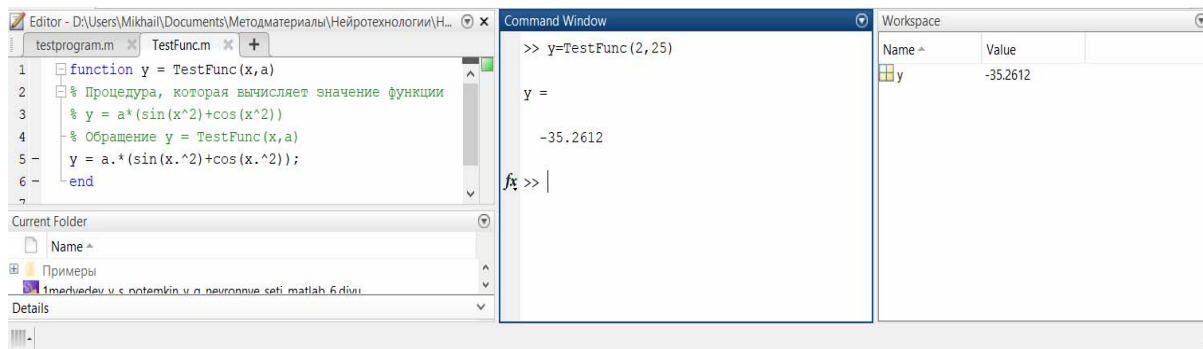


Рис. 1.26. Выполнение функции из командной строки

MatLab позволяет достаточно гибко оперировать с расчетами значений функций в диапазонах входных параметров. В следующем примере (рис. 1.27) покажем, как можно получить сразу вектор всех значений указанной функции при разных значениях аргумента.

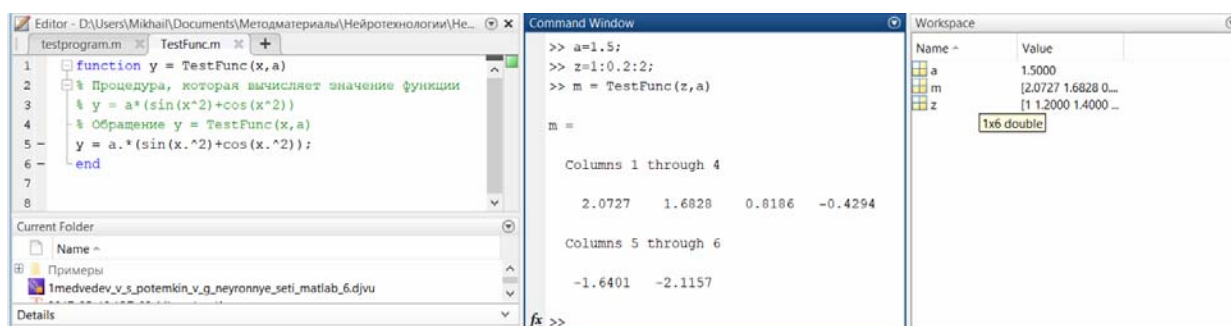


Рис. 1.27. Расчет вектора значений функции

Необходимо обратить внимание, что для того, чтобы функция могла работать с входными элементами в виде векторов и матриц необходимо указывать опцию поэлементного расчета – после имени переменной добавлять символ «.» (точку).

Работа с графиками

Рассмотрим описание нескольких графических функций системы MatLab.

Основной функцией для работы с графиками в MatLab является функция **plot**.

В случае, если подать на вход функции одномерный массив (вектор) y , то она построит кусочно-линейный график зависимости элементов y от их индексов.

В случае, если на вход поданы два вектора x и y , то команда ***plot(x,y)*** построит график зависимости y от x .

Построим график функции ***TestFunc*** из предыдущего пункта в интервале значений от $-\pi$ до π за 50 шагов. Для этого введем значения переменных так, как это показано на рис. 1.28.

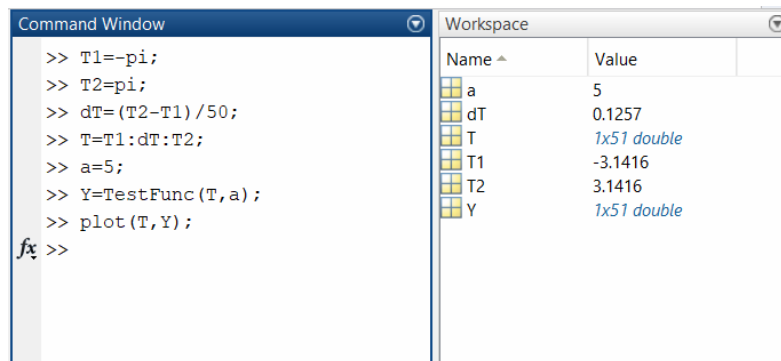


Рис. 1.28. Ввод интервала значений для построения графика

По команде ***plot(T,Y)*** строится график зависимости, который отображается в окне ***Figure 1*** (рис. 1.29).

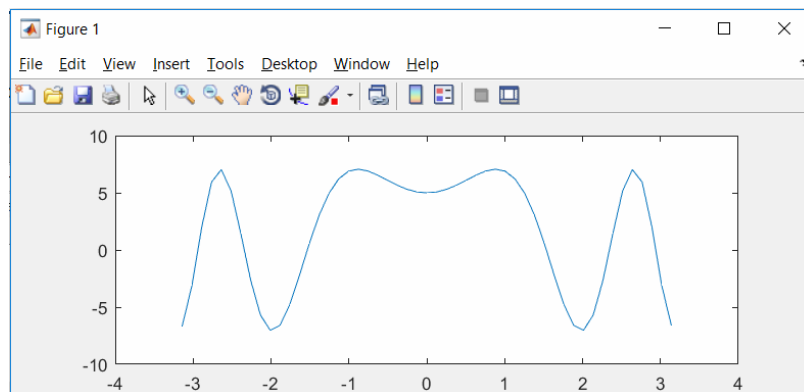


Рис. 1.29. График зависимости TestFunc в интервале от $-\pi$ до π

MatLab автоматически окрашивает каждый график в свой цвет (если это не задано пользователем), а также выделяет цветом разные значения на трехмерных графиках и поверхностях.

Применение команды ***hold on*** позволяет сохранить фокус вывода на ранее созданном графике и добавлять туда новые кривые.

Если необходимо вывести множество графиков в одно окно то применяется функция ***subplot***.

Для работы с трехмерными графиками используются функции **mesh** (сетчатая поверхность) и **surf** (гладкая поверхность).

На рис. 1.30 и 1.31 приведён пример совместного применения функций **subplot**, **mesh** и **surf** для построения графика трехмерной поверхности $a \cdot \sin(T)$ при $a=[0, 2]$ и $T=[-\pi, \pi]$.

```
a=0:0.2:2;  
T=-pi:pi/20:pi;  
S=a*sin(T);  
subplot(1,2,1); mesh(S);  
subplot(1,2,2); surf(S);
```

Рис. 1.30. Формирование двумерной поверхности

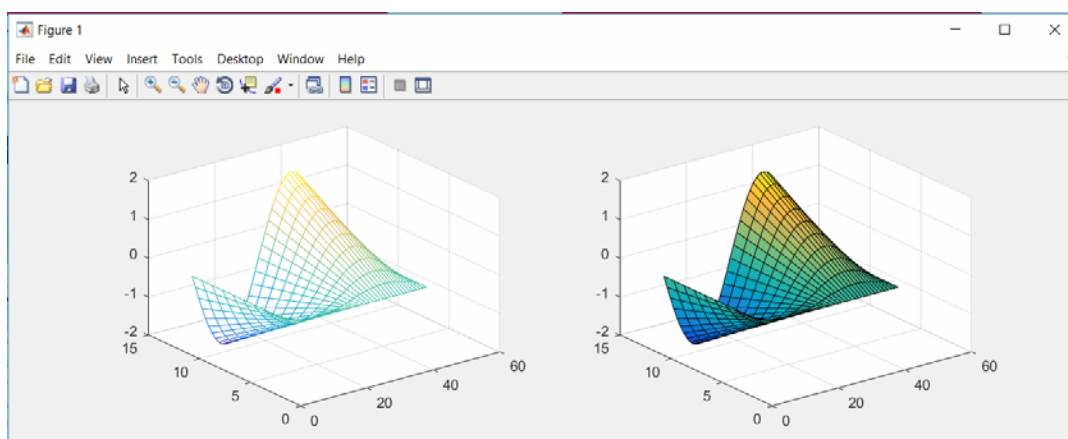


Рис. 1.31. Сетчатая (слева) и гладкая (справа) поверхности

Обратите внимание, что для корректной работы матричного выражения вектор \mathbf{a} необходимо транспонировать при помощи символа «'» (штрих).

Порядок выполнения работы

После выбора номера варианта выполнить следующие этапы:

1. Выбрать функцию для расчета в соответствии с вариантом.
2. Рассчитать значение заданной функции в точке x_1 .
3. С использованием оператора двоеточия сформировать вектор \bar{x} со значениями от x_{\min} до x_{\max} с шагом dx .
4. Для каждого элемента вектора \bar{x} вычислить значение функции, заданной по варианту, и записать результат в переменную y .
5. Используя созданные вектора построить график функции, подписать оси.

Индивидуальные задания

Для расчета значений натурального логарифма, тангенса и экспоненты в MatLab предусмотрены функции \log , \tan и \exp соответственно.

Номер варианта	Функция	x_1	x_{\min}	x_{\max}	dx	Контрольный вопрос
1	$y = \sin(3x)^2 - \cos(6x) + 1$	$\frac{\pi}{3}$	-2π	4π	$\frac{\pi}{30}$	1
2	$y = \frac{\sin(0,8x)^2 + \cos(-2,4x)}{2x}$	$\frac{\pi}{3}$	-2π	4π	$\frac{\pi}{30}$	3
3	$y = -6\cos(x^2) + 1,2x - 2x\sin(0,8x)$	$\frac{\pi}{3}$	-2π	4π	$\frac{\pi}{30}$	4
4	$y = \frac{\operatorname{tg}(3x)\sin\left(\frac{x}{3}\right)}{x^2 + 2}$	$\frac{\pi}{3}$	-2π	4π	$\frac{\pi}{30}$	5
5	$y = 5\sin\left(\frac{x}{4} + 2\right)^2 + \cos\left(\frac{x^3}{8} - 2x\right)$	$\frac{\pi}{3}$	-2π	4π	$\frac{\pi}{30}$	6
6	$y = \frac{2x^3 + 3x - 15}{8x^2 - 5x + 3}$	6	-9	9	0,09	7
7	$y = \frac{-2x^2 + 0,8x + 4}{3x^3 - 12x + 15}$	6	-9	9	0,09	8
8	$y = \frac{2x^4 - 8x^2 + 12}{-7x^3 - 3x + 5}$	6	-9	9	0,09	9
9	$y = \frac{x^2 - 5x + 8}{2x^2 - 3x + 6}$	6	-9	9	0,09	1
10	$y = \frac{x^2 + 5x - 12}{2x^3 - x}$	6	-9	9	0,09	2
11	$y = \frac{e^{0,2x}}{x} + 2x\sin(x + 2)$	8	3	10	0,07	3
12	$y = 3e^{-(x-1)^2} + 4e^{-(x-2)^2}$	8	3	10	0,07	4
13	$y = -2xe^{-(x-3)^2} + 3xe^{-(x-5)^2}$	8	3	10	0,07	5
14	$y = \ln(0,3x)x\sin(0,3x) + e^{-(x-3)^2}$	8	3	10	0,07	6
15	$y = \frac{x\ln(0,25x) + 12}{x\ln(0,25x)^2 + 0,8}$	8	3	10	0,07	7

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Представить программу, написанную в среде MatLab, и результаты ее работы.
4. Выводы.

Контрольные вопросы и задания

1. Как организована рабочая область MatLab?
2. Как осуществляется формирование переменной в MatLab?
3. Какие типы переменных существуют в MatLab?
4. Как осуществляется импорт и экспорт переменных?
5. Каким образом задается функция в MatLab?
6. Сколько значений может вернуть функция?
7. Средства предоставляет MatLab для работы с графиками?
8. Какие используются опции для форматирования графиков в MatLab?
9. Какой набор данных необходим для построения трехмерного графика?

Лабораторная работа 2

КЛАССИФИКАЦИЯ С ПОМОЩЬЮ ПЕРСЕПТРОНА

Цель работы: изучение модели нейрона персептрона и архитектуры персептронной однослойной нейронной сети; создание и исследование моделей персептронных нейронных сетей в системе MatLab.

2.1. Нейронные сети: основные положения

Основу каждой искусственной нейронной сети (ИНС) составляют относительно простые, в большинстве случаев – однотипные, элементы (ячейки), имитирующие работу нейронов нервной системы живых существ. Здесь и в дальнейшем под термином «нейрон» будем понимать искусственный нейрон, т. е. ячейка ИНС. Функционирование нейрона определяется его текущим состоянием, подобно тому, как это происходит в нервной системе, где нервные клетки могут находиться в возбужденном или заторможенном состоянии. У каждого нейрона есть группа синапсов – входных связей, соединенных с выходами других нейронов, ядро нейрона составляет ячейка нелинейного преобразователя, а на выходе – аксон, с которого сигнал (возбуждения или торможения) поступает на входные синапсы следующих нейронов. Общий вид нейрона приведен на рис. 2.1.

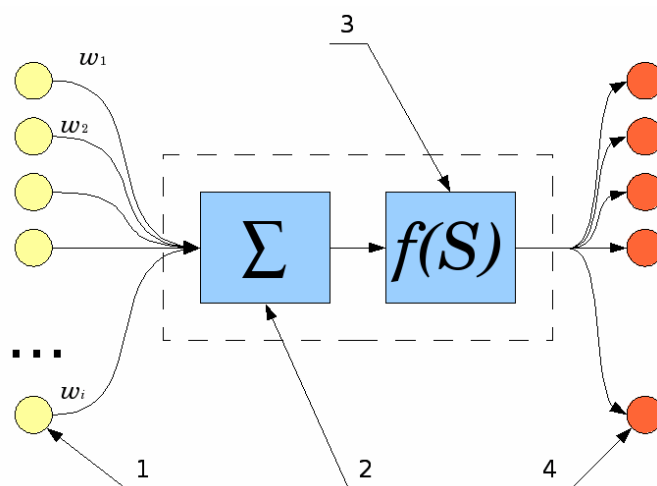


Рис. 2.1. Искусственный нейрон

Функционирование синапсов I (рис. 2.1) характеризуется величиной синаптической связи или весом синапса w_i . Синапс вычисляет

произведение входного сигнала на синаптический вес и передает полученное произведение на выход. По физическому смыслу вес синапса эквивалентен электрической проводимости, это довольно точная физическая аналогия с процессами, происходящими в нервной системе живых организмов.

Далее сигналы попадают на вход сумматора 2 (рис. 2.1). Соответственно, текущее состояние нейрона определяется, как взвешенная сумма его входов:

$$S = \sum_{i=1}^n w_i s_i . \quad (2.1)$$

Следующий элемент нейрона – нелинейный преобразователь 3 (рис. 2.1). Соответственно, выход нейрона есть функция его состояния:

$$y = f(s). \quad (2.2)$$

Функция нелинейного преобразователя называется функцией активации нейрона или его передаточной функцией. Функция активации может иметь различный вид, как показано на рис. 2.2. Вид функции во многом определяет способность ИНС решать прогностические и классификационные задачи при малом числе элементов. Кроме того, как будет видно в следующих главах, существуют ограничения в применимости некоторых функций активации в ИНС к наиболее эффективным методам обучения на базе алгоритмов обратного распространения ошибки.

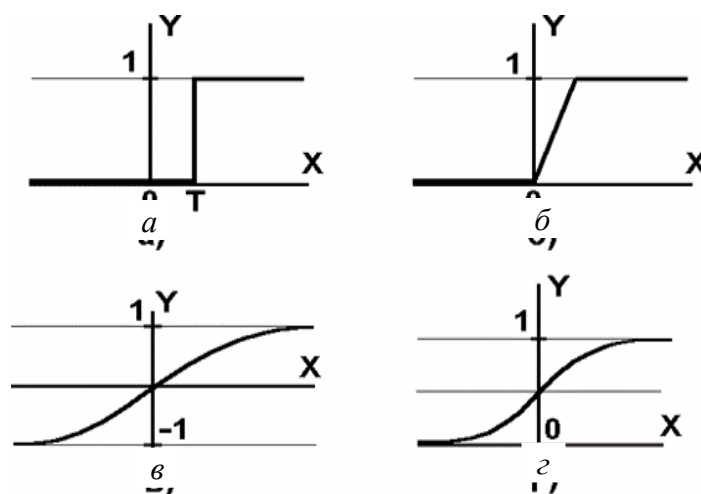


Рис. 2.2. Функции активации нейронов
 а – функция единичного скачка; б – линейный порог
 (гистерезис); в – сигмоида – гиперболический тангенс;
 з – сигмоида – формула (2.3)

Одной из наиболее распространенных является нелинейная функция с насыщением, так называемая логистическая функция или сигмоида (т. е. функция S -образного вида):

$$f(x) = \frac{1}{1 + e^{-\alpha x}}. \quad (2.3)$$

При уменьшении α сигмоида становится более пологой, в пределе при $\alpha = 0$ вырождаясь в горизонтальную линию на уровне 0,5, при увеличении α сигмоид приближается по внешнему виду к функции единичного скачка с порогом T в точке $x = 0$. Из вида функции (2.3) очевидно, что значение выходного сигнала нейрона лежит в диапазоне $[0,1]$. Важным преимуществом сигмоидной функции активации является ее дифференцируемость на всей области определения, что необходимо как раз для применения алгоритма обратного распространения ошибки и разработанных на его основе градиентных методов обучения ИНС.

Основная особенность любой ИНС – параллельная обработка сигналов. Именно за счет так называемого «мелкозернистого параллелизма», позволяющего одновременно на разных процессорах (разных ядрах процессора) выходные сигналы элементов достигается вычислительная эффективность ИНС. Степень эффективности ИНС такова, что до второго десятилетия XXI века суперкомпьютеры с тактовыми частотами в гигагерцы уступали в решении целого ряда задач мозгу человека, работающему на тактовой частоте не более десяти герц. Параллелизм ИНС достигается путем объединения большого числа нейронов в слои и соединения определенным образом нейронов различных слоев, а также, в некоторых конфигурациях, и нейронов одного слоя между собой. Расчет выходных сигналов элементов ИНС (синапсов, сумматоров, нелинейных преобразователей) выполняется послойно.

Рассмотрим архитектуру простейшей ИНС на примере однослойной трехнейронной персептронной сети (рис. 2.3). На n входов поступают сигналы, преобразующиеся в синапсах и поступающие на сумматоры на 3 нейрона, образующие единственный слой этой ИНС и выдающие три выходных сигнала:

$$y_j = f \left[\sum_{i=1}^n x_i w_{ij} \right], \quad j = 1 \dots 3. \quad (2.4)$$

Из формулы (2.4) видно, что веса синапсов одного слоя нейронов можно представить в виде матрицы W , в которой каждый

элемент w_{ij} соответствует весу i -й синаптической связи j -го нейрона. В этом случае срабатывание ИНС можно записать в матричном виде:

$$Y = F(XW), \quad (2.5)$$

где X и Y – соответственно вектора входных и выходных сигналов ИНС; $F(V)$ – функция активации, применяемая поэлементно к компонентам вектора V , который, в свою очередь, является результатом произведения вектора входных сигналов X на матрицу весов синапсов W .

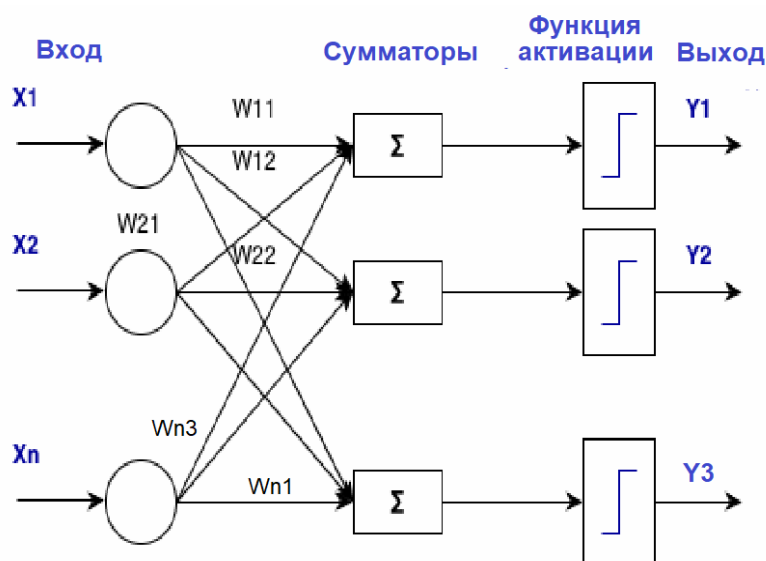


Рис. 2.3. Однослойный персептрон

В теории никаких ограничений на количество слоев ИНС и число нейронов в каждом слое не существует. На практике же размер ИНС ограничен ресурсами вычислительной системы – объемом памяти, количеством вычислительных ядер, архитектурой специализированной микросхемы, на которых может быть реализована ИНС.

При выборе архитектуры ИНС инженер каждый раз решает проблему:

- 1) рост сложности ИНС делает ее более «умной», повышает объем хранимой информации и сложность решаемых задач;
- 2) повышение сложности требует большего объема вычислительных ресурсов, а кроме того – больших объемов обучающих выборок для обучения большего числа элементов ИНС.

Выбор архитектуры ИНС производится исходя из особенностей и сложности задачи. Для целого ряда типовых задач выработаны оптимальные конфигурации ИНС. Если же задача не сводится ни к од-

ному из стандартных типов, разработчик вынужден синтезировать новую архитектуру, что является сложной задачей.

Как видно из вышеизложенного, сущность действий ИНС заключается в преобразовании входных сигналов в выходные с учетом весов синапсов. Из этого следует, что для решения определенной задачи необходимо подобрать оптимальные значения весов (и других обучаемых параметров, если они есть). Подбор значений всех переменных (они также называются обучаемыми) параметров ИНС с целью минимизация ошибки в решении заданных задач называется обучением. Главным образом обучаемыми параметрами являются веса синапсов, дополнительно могут подстраиваться параметры функций активации, например значение α в сигмоидной функции (2.3). При этом некоторые параметры, в том числе и часть синаптических связей могут оставаться неизменными.

От того, насколько качественно будет выполнен этап обучения ИНС, зависит способность сети решать поставленные перед ней задачи в процессе эксплуатации. Как и любая задача оптимизации, качество обучения и его трудоемкость (а значит и длительность) связаны обратной зависимостью – чем выше требуется качество обучения, тем больше будет трудоемкость. Сочетание этих двух параметров приходится выбирать на основе компромисса.

Выделяют два режима обучения ИНС – «с учителем» или «без учителя».

В первом варианте сети предъявляются пары значений «Входной сигнал» – «Требуемый выходной сигнал». Обучение, как уже говорилось, в этом случае заключается в подстройке обучаемых параметров с целью минимизации ошибки – несоответствия между расчетным ответом ИНС на входной сигнал и требуемым выходным сигналом.

Во втором случае «Требуемый выходной сигнал» не задается, выходы ИНС формируются самостоятельно, а обучаемые параметры изменяются сетью по алгоритму, учитывающему только входные и производные от них сигналы.

2.2. Однослойный персептрон

Однослойный персептрон ИНС, состоящая из одного слоя нейронов, каждый из которых имеет пороговую функцию активации (рис. 2.2, а).

Персептрон представляет собой простейшую нейронная сеть, параметры которой могут быть настроены на решение задачи классификации входных векторов.

Однослойный персептрон был изобретен в 1957 году Марком Розенблаттом и реализован в виде электронной машины «Марк-1» в 1960 году. Персептрон, таким образом, стал одной из первых реализованных ИНС, а «Марк-1» – первым в мире аппаратным нейрокомпьютером. Персептрон имеет простой алгоритм обучения, не связанный с дифференцированием, и способен решать лишь самые простые задачи. Эта модель вызвала так называемую «первую волну нейроинформатики» и стала толчком к развитию искусственных нейронных сетей.

Персептроны строятся на базе нейронов с пороговой функцией активации (реализована в MatLab под названием *hardlim*) с жесткими ограничениями.

При выполнении такта функционирования персептрона каждое значение элемента входного вектора умножается на соответствующий синаптический вес, затем эти значения суммируются результат поступает на вход функции активации. Если значение входа функции активации $n \geq 0$, то нелинейный преобразователь персептрона возвращает 1, если $n < 0$, то 0.

Использование пороговой функции активации позволяет персептрону классифицировать входные вектора, разделяя пространство входов на две области. Пространство входов делится на две области разделяющей гиперплоскостью (в случае двумерного пространства входов это линия, в случае трехмерного – плоскость). Эта гиперплоскость ортогональна вектору весов w и смещена на величину b . Точки, соответствующие входным векторам выше разделяющей гиперплоскости (например, линии), соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1. Те, которые расположены ниже соответствуют выходу персептрона, равному 0.

Изменение значений обучаемых параметров – синаптических весов и смещений – граница разделяющей гиперплоскости изменяет свое положение.

Персептрон без смещения всегда формирует разделяющую гиперплоскость, проходящую через начало координат; добавление смещения позволяет сформировать гиперплоскость, которая не проходит через начало координат.

В общих словах, работа персептрона заключается в классификации (отнесении) входных n -мерных сигналов к одному из некоторого числа классов. Как видно из предыдущих абзацев, с математической точки зрения решение этой задачи описывается путем разбиения n -мерного гиперпространства $(n-1)$ -мерными гиперплоскостями (рис. 2.4). Двумерное пространство (плоскость) делится линиями, трехмерное пространство – двумерной плоскостью и т. д. В случае однослойного персептрона это можно записать так:

$$\sum_{i=1} x_i w_{ik} = T_k, k = 1 \dots m. \quad (2.6)$$



Рис. 2.4. Геометрическая интерпретация однослойного персептрона

Каждая полученная при таком делении область является областью определения одного из классов выбранной классификации. Число таких классов для одной персептронной ИНС не превышает 2^m , где m – число выходов сети.

Следует оговориться, что существуют задачи, принципиально не решаемые персептронной ИНС. Это задачи, относящиеся к категории «линейно неразделимых».

Классический пример такого рода – решение «задачи исключаящего ИЛИ». Однослойный персептрон, состоящий из одного нейрона с двумя входами, не способен разделить плоскость (двумерное гиперпространство) на две полуплоскости так, чтобы осуществить классификацию входных сигналов на классы А и В (табл. 2.1).

Таблица 2.1

Исходные данные для задачи «Исключающего ИЛИ»

		X_2	
		0	1
X_1	0	A	B
	1	B	A

Уравнение сети для этого случая

$$x_1 w_1 + x_2 w_2 = T. \quad (2.7)$$

Это уравнение описывает прямую (одномерную гиперплоскость), которая ни при каких условиях не может разделить плоскость так, чтобы точки из множества входных сигналов, принадлежащие разным классам, оказались по разные стороны от прямой (рис. 2.5).

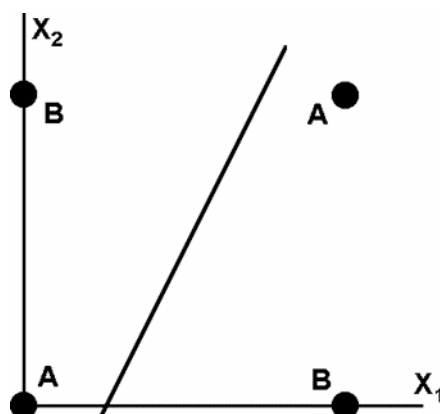


Рис. 2.5. Геометрическая интерпретация работы персептрона

Рассмотрев табл. 2.1 можно заметить, что данное разбиение на классы соответствует таблице истинности логической функции исключающего ИЛИ (XOR) для входных сигналов. Принципиальная линейная неразрешимость данной задачи однослойным персептроном получила в нейроинформатике название «проблемы исключающего ИЛИ» («The XOR Problem in Neural Networks»).

В нейроинформатике решение задач, попадающих в этот класс, заключается в увеличении числа слоев ИНС либо в применении сетей с нелинейными синапсами. Что, впрочем, тоже не всегда обеспечивает корректную классификацию.

Рассмотрим обучение ИНС на примере персептрона, представленного на рис. 2.3.

1. Проинициализировать элементы весовой матрицы (обычно небольшими случайными значениями).

2. Подать на входы поочередно в случайном порядке один из входных векторов, которые сеть должна научиться различать, и вычислить ее выход.

3. Если выход правильный, перейти на шаг 4.

Иначе вычислить разницу между целевым (Y_T) и полученным значениями выхода:

$$\delta = Y_T - Y. \quad (2.8)$$

Подстроить веса синапсов в соответствии с формулой

$$w_{ij}(t+1) = w_{ij}(t) + v \cdot \delta \cdot x_i, \quad (2.9)$$

где t и $t + 1$ – номера соответственно текущего и следующего шагов обучения; v – коэффициент скорости обучения, $0 < v < 1$; i – номер входа; j – номер нейрона в слое.

Очевидно, что если $Y_T > Y$ весовые коэффициенты будут увеличены и тем самым уменьшат ошибку. В противном случае они будут уменьшены, и Y тоже уменьшится, приближаясь к Y_T .

4. Цикл с шага 2, до тех пор, пока сеть не перестанет ошибаться.

В силу случайного характера предъявления ИНС примеров обучающей выборки данный алгоритм не только не обеспечивает сходимости за гарантированное число итераций, но и в принципе не всегда гарантирует полный успех обучения ИНС.

2.3. Классификации однослойным персептроном на 2 класса

Обучим однослойную ИНС из одного нейрона с пороговой функцией активации – однослойный персептрон – классификации двумерных входных векторов на два класса.

Построение обучающей выборки.

Каждый из пяти векторов-столбцов в переменной X определяет 2-мерные входные векторы, а вектор строки T определяет целевые («правильные» в терминологии обучения) классы для этих векторов.

Визуализировать вектора в MatLab можно при помощи команды *plotpv* (рис. 2.6, 2.7).

```
% X - входные вектора
X = [-0.5 -0.5 -0.6 0.3 0.1;-0.5 0.5 0.0 0.5 1.0];
% T - целевые классы для входных векторов
T = [1 1 1 0 0];
% визуализация обучающей выборки
plotpv (X, T);
```

Рис. 2.6. Подготовка и визуализация данных для обучения персептрона

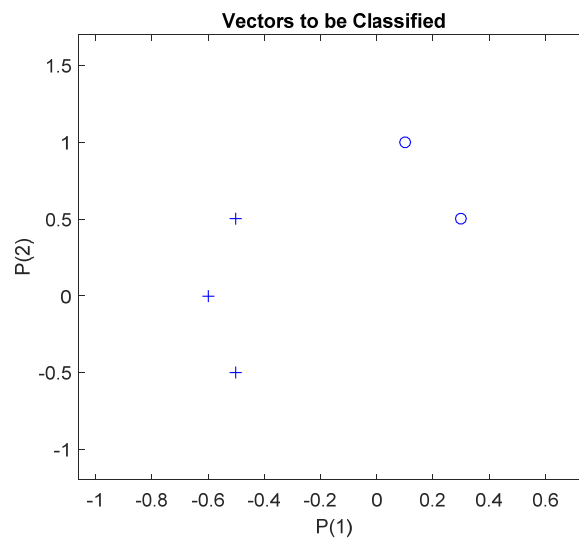


Рис. 2.7. Результат визуализации обучающей выборки

Архитектура сети

Персептрон должен классифицировать 5 входных векторов в X на два класса, заданных переменной T . Персептроны строятся на базе нейронов с пороговой функцией активации (*hardlim*). Такие нейроны, как уже было сказано выше, способны разделять пространство входных векторов прямой линией на два класса (0 и 1).

Команда *perceptron* создает новую ИНС с одним нейроном (рис. 2.8) В ней X_1 , X_2 – координаты входных векторов, Y -выход (класс). Затем сеть обучается на предъявленных данных, мы также можем проверить ее начальные значения синаптических весов и смещения. (Обычно этап настройки можно пропустить, так как он автоматически выполняется командами *adapt* или *train*.).

Далее приведем фрагмент кода MatLab (рис. 2.9), в котором создается и настраивается собственно нейронная сеть, а затем предпринимается попытка ее визуализации при помощи команды *plotpc*.

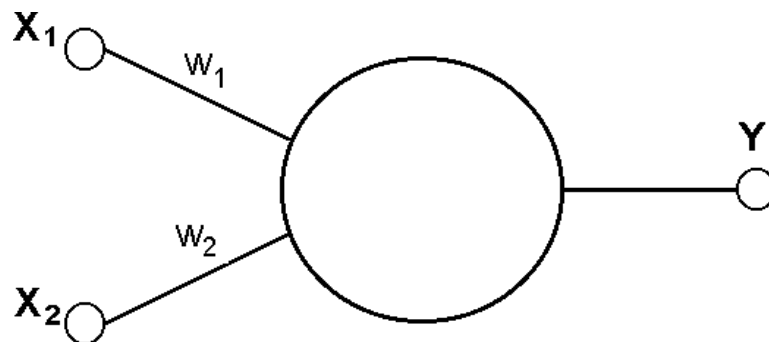


Рис. 2.8. Однослойный персептрон с 1 нейроном

```
% Создание ИНС
net = perceptron;
% Настройка ИНС для работы с данными
net = configure(net,X,T);
% визуализация обучающей выборки
plotpv(X,T);
% визуализация разделяющей линии персептрона
plotpc(net.IW{1},net.b{1});
```

Рис. 2.9. Создание и настройка персептрона

Отметим, что команда *net=configure(net,x,t)* принимает входные данные *x* и целевые данные *t* и настраивает входы и выходы сети в соответствии с ними. Вектор входов определяет размерность входного пространства сети, вектор выходов – размерность выходного пространства и (в случае однослойной ИНС) количество нейронов, а также размерность пространства синаптических весов. Отметим также, что это не единственный вариант применения этой команды.

В результате выполнения этого фрагмента программы визуализация персептрона (команда *plotpc*) не изменяет изображения, оно остается таким же, как на рис. 2.7. Это происходит из-за того, что начальные веса синапсов установлены на ноль, поэтому линия классификации не появляется на графике. Эта ситуация будет меняться по мере обучения сети.

В приведенном на рис. 2.10 фрагменте кода входные и целевые данные преобразуются в последовательности (массив ячеек, где каждый столбец указывает временной шаг) и копируются три раза при помощи команды *repmat*, чтобы сформировать серии *XX* и *TT*.

Команда *adapt* обновляет сеть для каждого временного шага в серии и возвращает новый сетевой объект, который выполняет функции лучшего классификатора для имеющихся данных.

```
% создание серии для входных данных
XX = repmat(con2seq(X),1,3);
% создание серии для выходных данных
TT = repmat(con2seq(T),1,3);
% настройка ИНС путем предъявления серий
net = adapt(net,XX,TT);
% повторная визуализация разделяющей линии персептрона
plotpc(net.IW{1},net.b{1});
```

Рис. 2.10. Настройка ИНС на решение задачи классификации

Видно (рис. 2.11), что полученная разделительная линия, задающаяся параметрами персептрона – весами синапсов и смещением – прошла таким образом, что надежно оделила облако точек нулевого класса от точек первого класса. Это говорит о том, что в результате всего трех тактов обучения ИНС, содержащая всего один персептрон в одном слое, обучилась надежно классифицировать линейно разделимые классы в двумерном пространстве входов.

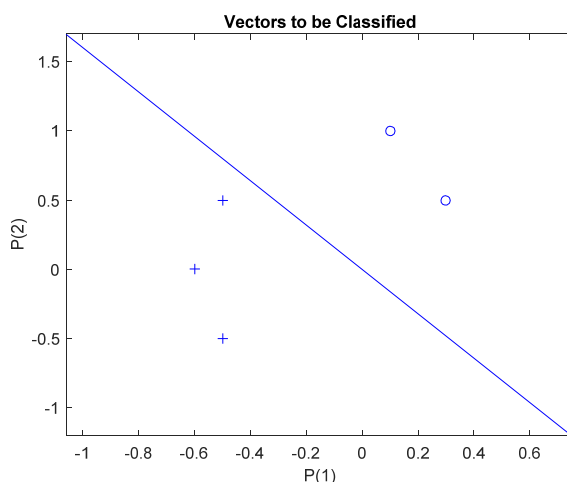


Рис. 2.11. Обучающая выборка и линия классификации

Моделирование работы ИНС

Теперь вызовем нашу ИНС в режиме решения предъявленного примера для классификации входного вектора, не входившего в обучающую выборку, например $[0,5; 1,3]$ (рис. 2.12, 2.13). Изображение этой точки, наложенное на исходную обучающую выборку, продемонстрирует работу обученной ИНС по решению задачи.

```
% задание тестовой точки x
x = [0.5; 1.3];
% занесение в переменную y результата вызова сети с входным
вектором x
y = net(x);
% нанесение на график результата работы сети
plotpv(x,y);
% выделение точки для окрашивания
point = findobj(gca,'type','line');
% окрашивание точки в красный цвет
point.Color = 'red';
% включаем режим вывода на тот же график
hold on;
% вывод точек обучающей выборки
plotpv(X,T);
% вывод разделяющей линии
plotpc(net.IW{1},net.b{1});
% выключаем режим вывода на тот же график
hold off;
```

Рис. 2.12. Тестирование ИНС на новом примере и визуализация теста

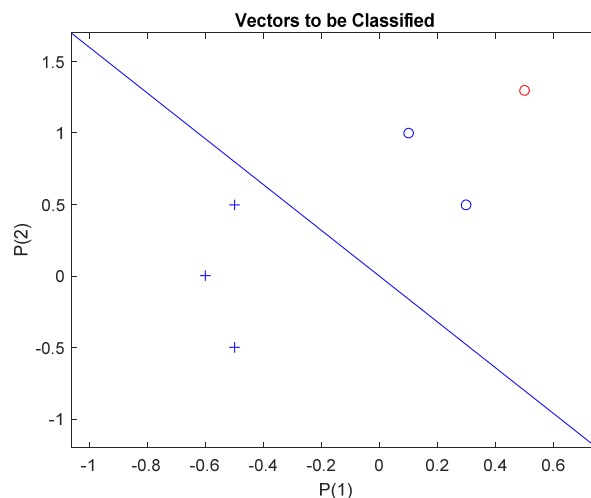


Рис. 2.13. Результат классификации тестового примера

Для того чтобы она отличалась от точек обучающей выборки, закрасим ее красным цветом. Затем включим режим *hold*, чтобы предыдущий график не был стерт, и нанесем на график точки обучающей выборки и линию классификации. Видим, что персептрон правильно классифицировал нашу новую точку (выделена красным), отнеся ее к нулевому классу (обозначен кружком), а не к первому (обозначен плюсом). Соответствующий код – на рис. 2.12, результат визуализации – на рис. 2.13.

2.4. Классификация однослойным персептроном на 4 класса

Постановка задачи

Решим задачи классификации примеров в двумерном пространстве на 4 класса. Очевидно (ответьте себе на вопрос, почему это так), что для решения этой задачи ИНС из одного нейрона недостаточно. Поэтому в примере, который мы рассмотрим ниже, строится персептрон, имеющий два входа, два выхода и содержащий 2 нейрона с пороговой функцией активации (рис. 2.14).

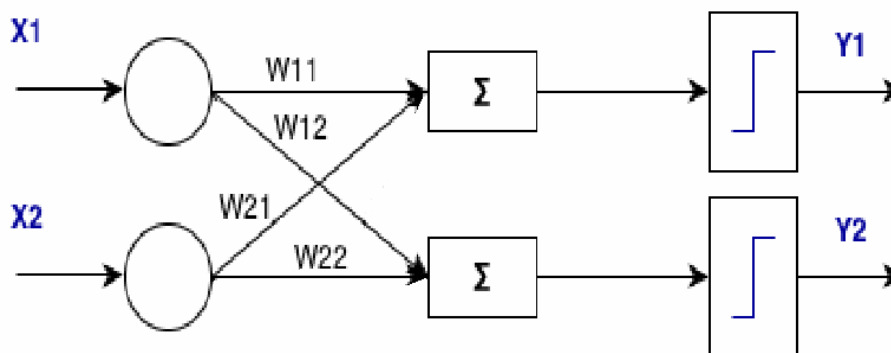


Рис. 2.14. Двухнейронный персептрон

Определение обучающих выборок

Сгенерируем случайным образом обучающие выборки по 15 примеров для 4 классов, которые в дальнейшем планируем разделять. Код для решения этой задачи приведен на рис. 2.15.

В результате выполнения этого кода строится график (рис. 2.16), содержащий облака точек для 4 линейно-разделимых классов. Проанализируйте код, ответьте на вопрос из чего следует их линейная разделимость?

```

close all, clear all, clc, format compact
% количество примеров в каждом классе
K = 15;
% задаем обучающие выборки для 4 классов
q = .6; % смещение классов относительно друг друга
A = [rand(1,K)-q; rand(1,K)+q];
B = [rand(1,K)+q; rand(1,K)+q];
C = [rand(1,K)+q; rand(1,K)-q];
D = [rand(1,K)-q; rand(1,K)-q];
% визуализация обучающих выборок
plot(A(1,:),A(2,:), 'bs')
hold on
grid on
plot(B(1,:),B(2,:), 'r+')
plot(C(1,:),C(2,:), 'go')
plot(D(1,:),D(2,:), 'm*')
% нанесение текстовых меток классов
text(.5-q,.5+2*q,'Класс A')
text(.5+q,.5+2*q,'Класс B')
text(.5+q,.5-2*q,'Класс C')
text(.5-q,.5-2*q,'Класс D')
% определение выходных кодировок классов
a = [0 1]';
b = [1 1]';
c = [1 0]';
d = [0 0]';

```

Рис. 2.15. Генерация обучающей выборки на 4 класса

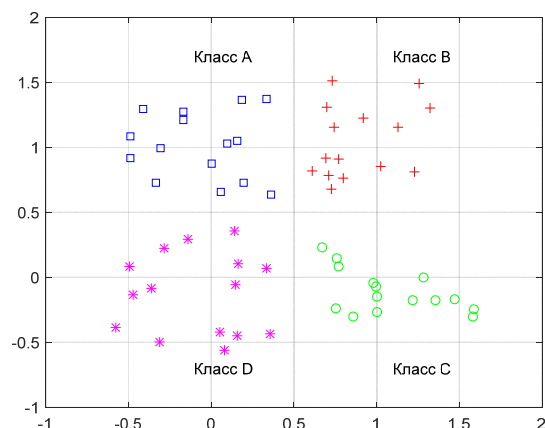


Рис. 2.16. Исходные данные для классификации на 4 класса

Разберитесь с переменными *a*, *b*, *c*, *d*. Почему их значение именно таково? Почему код, приведенный на рис. 2.17 будет неверным?

```
% Почему эта кодировка нерабочая?  
a = [0 0]';  
b = [1 1]';  
d = [0 1]';  
c = [1 0]';  
% Почему эта кодировка нерабочая?  
a = [0 1]';  
b = [1 1]';  
d = [1 0]';  
c = [0 1]';
```

Рис. 2.17. Неверные варианты кодировок

Подготовка исходных данных для обучения

Соберем из данных для всех четырех классов единую обучающую выборку. Для этого объединим исходные данные по классам *A*, *B*, *C* и *D* в один массив, а соответствующие им значения правильных ответов *T* получим, скопировав требуемое количество раз кодировку правильного класса при помощи команды *repmat*. Данный код приведен на рис. 2.18.

```
% Входные сигналы сформируем, объединив матрицы классов A, B,  
C, D  
P = [A B C D];  
% определим массив целевых векторов P, включив в него столько  
раз копию кодировки класса, сколько было в классе входных векторов  
T = [repmat(a,1,length(A)) repmat(b,1,length(B)) ...  
repmat(c,1,length(C)) repmat(d,1,length(D)) ];
```

Рис. 2.18. Обучающая выборка из входных *P* и выходных *T* данных

Генерация и обучение персептрона

Создадим персептрон в переменной *net* при помощи функции *perceptron* и затем обучим его в цикле *while*, повторяя процедуру *adapt* не более 1000 раз или пока ошибка *E* не станет равна 0. В каждом цикле обучения прорисовываются разделительные линии при помощи команды *plotpc*, так что будет наглядно видно, как они все лучше разделяют классы. Соответствующий код приведен на рис. 2.19.

```

% создаем персептрон в переменной net
net = perceptron;
% обучение персептрона в цикле
E = 1;
net.adaptParam.passes = 1;
linehandle = plotpc(net.IW{1},net.b{1});
n = 0;
while (sse(E) & n<1000)
n = n+1;
[net,Y,E] = adapt(net,P,T);
linehandle = plotpc(net.IW{1},net.b{1},linehandle);
drawnow;
end
% визуализируем структуру ИНС
view(net);

```

Рис. 2.19. Создание и обучение ИНС

Визуализация итогового состояния ИНС приведена на рис. 2.20. Видно, что разделительные линии корректно разделяют четыре класса на области.

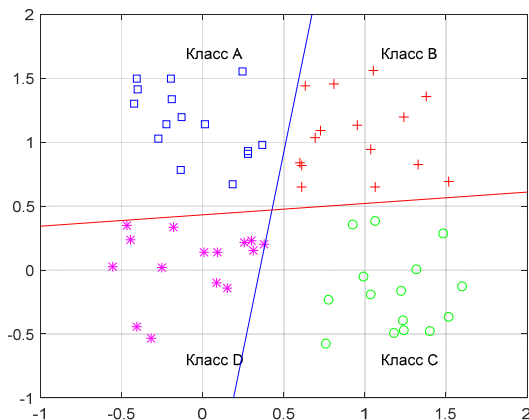


Рис. 2.20. Визуализация обученной ИНС

Завершающая команда `view` выводит на экран структуру ИНС (рис. 2.21). Видно, что она имеет два входа, два выхода и два нейрона с пороговой функцией активации (***Hard Limit***).

Использование обученной ИНС

Классифицируем пример, не входивший в обучающую выборку, например $[0.8; 1.3]$ (рис. 2.22).

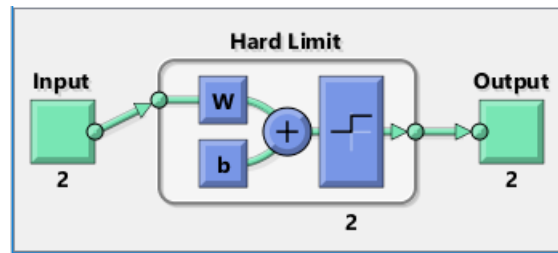


Рис. 2.21. Визуализация структуры ИНС

```

Command Window
>> p=[0.8; 1.3];
>> y=net(p)
y =
    1
    1
fx >> |

```

Рис. 2.22. Вызов ИНС на одиночный пример

Теперь получившийся отклик ИНС (вектор y) необходимо сравнить с кодировками классов a , b , c и d .

Порядок выполнения работы

1. Построить нейронную сеть, которая производит классификацию на заданное количество классов с помощью функции *perceptron*. Произвести начальную инициализацию весов нейронной сети с помощью функции *configure*.

2. Построить обучающую выборку, которая позволяет правильно классифицировать заданную проверочную выборку. Координаты точек обучающей выборки должны быть подобраны геометрически. Для этого первоначально нанести точки проверочной выборки на график, ориентировочно построить вокруг каждой из этих точек несколько точек обучающей выборки, затем записать координаты этих точек в массив обучающей выборки. На каждую из точек проверочной выборки должно приходиться не менее 3 точек обучающей выборки.

3. Показать обучающую выборку на графике с помощью функции *plotpv*.

4. Произвести обучение нейронной сети на составленной обучающей выборке с использованием функции *adapt*. В процессе обуче-

ния показывать изменение линий разбиения на классы в нейронной сети с помощью функции *plotpc*.

5. Используя обученную нейронную сеть, произвести классификацию индивидуального для каждого варианта проверочного множества. Поскольку номер класса для каждой точки может меняться при каждом новом обучении сети, номера классов в таблице выведены условно. Точки с разными номерами классов в таблице при классификации должны иметь разные номера классов, точки с одинаковыми – одинаковые. Привести график классификации точек (с помощью функций *plotpc*, *plotpv*). Вывести результаты работы сети на график. Для графика возможно применение различных цветов (команды *findobj*, *set*).

Индивидуальные задания

Таблица 2.2

Индивидуальные задания

№ вв	Количество классов для классификации	Координаты точек проверочного множества и номер класса, к которому принадлежит каждая точка	Контрольный вопрос
1	3	[0;0]–1; [1;1]–2; [–1;–1]–3	1
2	4	[2;1]–1; [1;0]–2; [0;–1]–3	2
3	3	[0;0]–1; [1;1]–1; [–1;–1]–3	3
4	4	[1;0]–1; [–1;1]–2; [–1;–1]–2	4
5	3	[0;0]–1; [1;1]–2; [–1;–1]–3	5
6	4	[0;0]–1; [1;1]–1; [–1;–1]–1	6
7	3	[–1;0]–1; [–1;1]–2; [–1;–1]–3	7
8	4	[0;1]–1; [1;–1]–2; [–1;–1]–3	1
9	3	[2;0]–1; [1;1]–2; [–1;1]–2	2
10	4	[0;0]–1; [1;1]–2; [–1;–1]–1	3
11	3	[0;0]–1; [1;1]–1; [–1;–1]–2	4
12	4	[0;0]–1; [1;1]–2; [–1;–1]–3	5
13	3	[0;0]–1; [1;1]–1; [–1;–1]–1	6
14	4	[0;1]–1; [1;–1]–2; [–1;–1]–3	7

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.

3. Теоретические сведения.
4. Представить графическое и табличное представление обучающего множества. Показать их линейную разделимость.
5. Представить графическое и табличное представление проверочных точек.
6. Представить программу, написанную в среде MatLab, и результаты классификации.
7. Выводы.

Контрольные вопросы

1. Структура персептронного нейрона.
2. Правило нахождения количества нейронов в персептроне для распознавания заданного числа классов.
3. Построение линий классификации персептрона на основании его весов.
4. Процесс обучения персептрона.
5. Параметры функций `perceptron`, ***adapt***, ***plotpc***, ***plotpv***.
6. Анализ информации, выдаваемой функцией ***display***.
7. Тестирование сети на примерах, не входящих в обучающую выборку.

Лабораторная работа 3
**АППРОКСИМАЦИЯ ФУНКЦИИ ПРИ ПОМОЩИ
ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ**

Цель работы: изучение возможности аппроксимации с помощью нейронной сети прямого распространения.

3.1. Постановка задачи аппроксимации

Постановка задачи аппроксимации

Аппроксимацией (приближением) функции $f(x)$ называется нахождение такой функции $g(x)$ (аппроксимирующей функции), которая была бы близка заданной. Разработано несколько критериев близости функций $f(x)$ и $g(x)$.

Основной задачей аппроксимации является построение аппроксимирующей функции, наиболее близко проходящей около заданных точек или около графика непрерывной функции. Возникновение такой задачи связано с наличием погрешностей в исходных данных. В этом случае точно проводить функцию через все точки исходных данных, как это делается в интерполяции, нецелесообразно.

Аппроксимация позволяет исследовать числовые характеристики и качественные свойства объекта, сводя задачу к изучению более простых или более удобных объектов (например, таких, характеристики которых легко вычисляются или свойства которых уже известны).

Близость исходной и аппроксимирующей функций определяется числовой мерой – критерием аппроксимации (близости). Наибольшее распространение получил критерий близости в заданных точках, равный сумме квадратов отклонений расчетных значений от «экспериментальных». Такой критерий называется «квадратичным»

$$R = \sum_{i=1}^n \beta_i (y_i - g_i)^2. \quad (3.1)$$

Здесь y_i заданные табличные значения функции в i -й точке; g_i – значения аппроксимирующей функции в i -й точке; β_i – весовые коэффициенты, задающие относительную важность i -й точки таким образом, что рост β_i при минимизации критерия R в первую очередь приводит к уменьшению отклонения в i -й точке.

Квадратичный критерий обладает рядом достоинств, в числе которых дифференцируемость и наличие единственного решения задачи аппроксимации при полиномиальных аппроксимирующих функциях.

Выделяют две основные задачи аппроксимации:

- 1) получение аппроксимирующей функции, описывающей имеющиеся данные, с погрешностью не выше заданной;
- 2) получение аппроксимирующей функции заданной структуры с наилучшей возможной погрешностью.

Чаще всего первая задача сводится ко второй перебором различных аппроксимирующих функций и последующим выбором наилучшей.

Нейросетевое решение задачи аппроксимации

Как было показано выше, аппроксимация может сводиться к нахождению функциональной зависимости по имеющемуся набору точек. В данном случае функциональная зависимость будет представлена в нейросетевом базисе, т. е. через комбинацию активационных функций нейронных сетей.

Алгоритм обратного распространения ошибки

Обучение многослойного персептрона чаще всего происходит с помощью алгоритма обратного распространения ошибки или его модификации.

Обратное распространение ошибки (*Back Propagation of Error*) – системный метод обучения ИНС. Он имеет серьезное математическое обоснование. Несмотря на некоторые ограничения, этот алгоритм сильно расширил область проблем, где могут быть использованы ИНС.

В основу метода положено обучение ИНС по образцам с помощью примеров обучающей выборки. Для каждого примера из обучающей выборки считается известным требуемое (целевое) значение выхода ИНС, то есть метод обратного распространения ошибки представляет собой обучение с учителем.

Такое обучение можно представить как решение оптимизационной задачи на минимизацию функции ошибки (невязки) E на всей обучающей выборке путем подстройки обучаемых параметров ИНС – значений весов синапсов w , параметров активационных функций и т. п.

$$E = \frac{1}{2} \sum_i^n (t_i - y_i)^2, \quad (3.2)$$

где t_i – требуемое (целевое) значение выхода на i -м образце выборки; y_i – реальное (рассчитываемое ИНС) значение; n – число образцов обучающей выборки.

Минимизация функционала ошибки E в методе обратного распространения ошибки осуществляется с помощью одного из градиентных методов оптимизации, например, «наискорейшего спуска» или его модификаций. При этом веса синапсов изменяются в направлении, обратном направлению наибольшего возрастания функции ошибки – в направлении антиградиента E :

$$W(t+1) = W(t) - \eta \frac{\partial E}{\partial W}, \quad (3.3)$$

где $0 < \eta \leq 1$ – «шаг обучения», определяемый пользователем параметр, определяющий скорость обучения, в ряде методов он подбирается итерационно.

Метод обратного распространения ошибки реализуют в двух вариантах. В первом варианте обучаемые параметры пересчитываются после подачи всей обучающей выборки, и ошибка имеет вид

$$E = \frac{1}{2} \sum_i^p (t_i - y_i)^2. \quad (3.4)$$

Во втором подходе ошибка пересчитывается после каждого примера:

$$E_i = \frac{1}{2} (t_i - y_i)^2, \quad (3.5)$$

$$E(W) = \frac{1}{2} \sum_{i,p} (t_i - y_i)^2. \quad (3.6)$$

Пусть

$$\Delta W_{ij}(t) = -\eta \frac{\partial E}{\partial W_{ij}}, \quad (3.7)$$

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}(t). \quad (3.8)$$

Тогда

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial y_{ij}} \cdot \frac{\partial y_i}{\partial S_i} \cdot \frac{\partial S_j}{\partial W_{ij}}, \quad (3.9)$$

$y_j(S_j)$ – активационная функция. Для

$$y_j = \bar{f}(S) = \frac{1}{1 + e^{-S_j}}, \quad (3.10)$$

$$\begin{aligned} \bar{f}'(S) &= \frac{\partial}{\partial S_j} \left(\frac{1}{1 + e^{-S_j}} \right) = \frac{1}{(1 + e^{-S_j})^2} (-e^{-S_j}) = \\ &= \frac{1}{1 + e^{-S_j}} * \frac{-e^{-S_j}}{1 + e^{-S_j}} = y_j(1 - y_j), \end{aligned} \quad (3.11)$$

$$\frac{dy_i}{dS_j} = (1 - y_j)^2; \quad (3.12)$$

рассмотрим третий сомножитель:

$$S_j = \sum W_{ij} \cdot y_i^{n-1}; \quad (3.13)$$

$$\frac{\partial S_i}{\partial W_{ij}} = y_i^{(n-1)}. \quad (3.14)$$

Можно показать, что

$$\frac{\partial E}{\partial y_i} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot W_{jk}^{(n+1)}, \quad (3.15)$$

при этом суммирование по k идет среди нейронов n -го слоя.

Введем новое обозначение:

$$\delta_j = \frac{\partial E}{\partial y_k} \cdot \frac{dy_j}{dS_j}, \quad (3.16)$$

$$\delta_j^{n-1} = \left[\sum_k \delta_k^n \cdot W_{jk}^n \right] \cdot \frac{dy_j}{dS_j}. \quad (3.17)$$

Для внутреннего нейрона (3.17)

$$\delta_j^n = (d_j^n - y_j^n) \cdot \frac{dy_j}{dS_j}. \quad (3.18)$$

Для внешнего нейрона (3.18)

$$\Delta W_{ij}^n = -\eta \delta_j^{(n)} \cdot y_j^{n-1}; \quad (3.19)$$

$$W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}. \quad (3.20)$$

С учетом формул (3.5)–(3.20) обучение по методу обратного распространения ошибки производится в соответствии со следующим алгоритмом:

1. Производится инициализация начальных значений обучаемых параметров. Обычно начальные значения составляют малые случайные величины.

2. На входы ИНС подается один из примеров обучающей выборки. В режиме обычного функционирования ИНС (сигналы распространяются от входов к выходам) производится расчет значений выходов всех нейронов.

3. Рассчитываются значения δ_j^n для нейронов выходного слоя по формуле для внешнего нейрона (3.18).

4. Рассчитываются значения δ_j^n для всех внутренних нейронов по формуле (3.17).

5. С помощью формулы (3.19) для всех связей вычисляются приращения весовых коэффициентов ΔW_{ij}^n .

6. Скорректировать веса синапсов по формуле (3.20): $W_{ij}(t+1) = W_{ij}(t) + \Delta W_{ij}$.

7. Повторить шаги 2–6 для каждого примера обучающей выборки до тех пор, пока значение функционала ошибки E не станет достаточно маленькой.

Альтернативным условием останова цикла на шаге 7 является невозможность обучить ИНС до приемлемой точности. Это решение принимается исходя из того, что приращения весовых коэффициентов ΔW_{ij}^n станут слишком малы и количество итераций в этой точке больше заданного.

Такая ситуация может возникнуть в следующих двух случаях:

1. Алгоритм оптимизации попал в локальный экстремум функционала ошибки E . Для того чтобы исключить данную возможность, необходимо стартовать обучение из другой стартовой конфигурации обучаемых параметров.

2. Архитектура ИНС недостаточна для решения задачи. Если многократный запуск обучения сети все-таки не привел к успешному обучению, то, скорее всего, следует усложнить сеть – увеличить количество нейронов или даже количество слоев.

3.2. Реализация многослойной сети в MATLAB

Согласно теореме, доказанной в 1989 г. Л. Фунахаши, произвольно сложная функция может быть аппроксимирована двуслойной нейронной сетью, имеющей произвольные активационные функции. Поэтому классически для решения задачи аппроксимации используют многослойный персептрон, причем, это могут быть как функции одной, так и нескольких переменных.

Аппроксимация гладкой функции одной переменной

В качестве примера рассмотрим функцию одной переменной $y = x^2 \sin\left(2x + \frac{\pi}{2}\right)$, зашумленную линейным шумом.

Сгенерированный набор точек будет выглядеть так, как это показано на рис. 3.1.

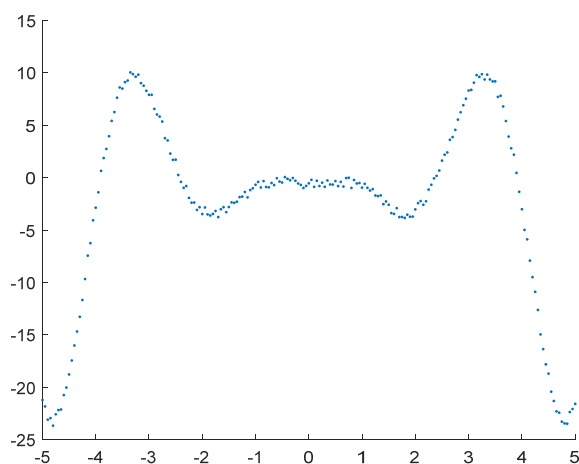


Рис. 3.1. График функции для обучения ИНС

Подготовка данных для обучения

Для генерации обучающей выборки используем программу на языке MatLab, показанную на рис. 3.2.

```
% количество точек
L=200;
% генерация значений аргумента X
X=-5:(10/L):5;
% расчет точек функции, зашумленной линейным шумом
Noise=0.5*rand(1,(L+1))*2-1.0;
Y=X.^2.*sin(2*X*pi/2)+Noise;
```

Рис. 3.2. Генерация исходных данных для обучения ИНС

Создание и обучение ИНС для аппроксимации функции

Создадим ИНС для аппроксимации сгенерированных данных при помощи команды *feedforwardnet*, обучим ее при помощи команды *train* а затем протестируем на новом диапазоне значений используя функцию *sim*, см. рис. 3.3. Разберитесь с синтаксисом этих команд.

```
hold on;
% визуализируем обучающую выборку
plot(X,Y,'LineStyle','none','Marker','.', 'MarkerSize',5);
% создаем ИНС
Net = feedforwardnet(10,'traingdx');
% обучаем ИНС
Net=train(Net,X,Y);
% Тестируем ИНС на интервале шире, чем обучающая выборка
X_rez=-10:0.1:10;
Y_rez = sim(Net, X_rez);
% визуализируем результаты аппроксимации
plot(X_rez,Y_rez,'LineStyle','none','Marker','.', 'MarkerSize',15);
hold off;
```

Рис. 3.3. Создание и обучение ИНС прямого распространения

В процессе обучения MatLab выводит интерфейсную форму нейросети (рис. 3.4). Видна архитектура системы – количество слоев (2) и нейронов в них, типы функций активации ('tansig' и 'purelin'), алгоритм обучения (модифицированный градиентный

спуск), тип функции ошибки (среднеквадратическая), а также параметры хода обучения.

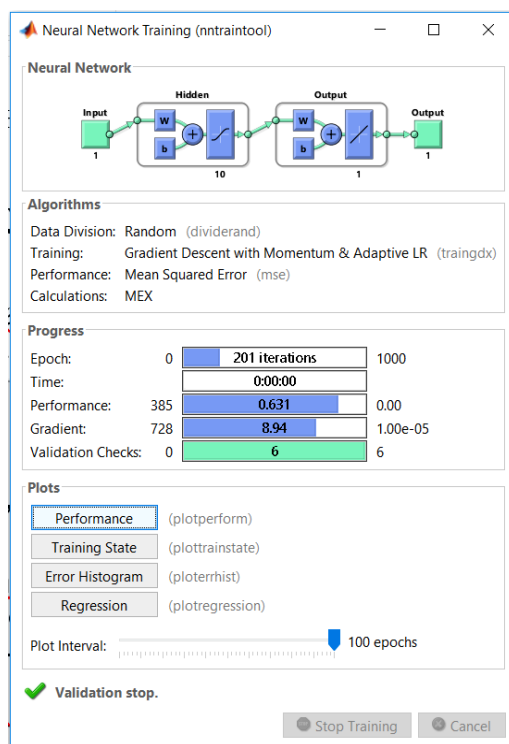


Рис. 3.4. Интерфейсная форма нейросети в MatLab

Кнопка Performance содержит график динамики погрешности сети по тактам обучения (рис. 3.5).

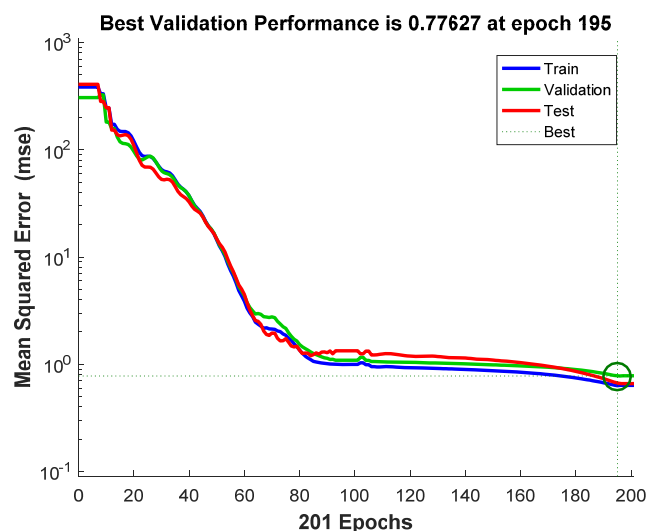


Рис. 3.5. График хода обучения

Останов процесса обучения по умолчанию происходит, когда значение функционала ошибки ИНС возрастает шесть эпох подряд.

В этом примере результат адекватен из следующих соображений:

1. Заключительная среднеквадратичная ошибка (MSE) мала.
2. Ошибка проверочного (Validation) и тестового (Test) наборов имеют подобные характеристики.
3. Переобучения не произошло (после точки останова происходит лишь увеличение MSE проверочного набора до 201-й эпохи).

На рис. 3.6 можем увидеть аппроксимированную функцию-результат расчета сети при помощи функции *sim*. Как это показано на рис. 3.3, тестирование сети проводится на более широком диапазоне, чем обучающая выборка.

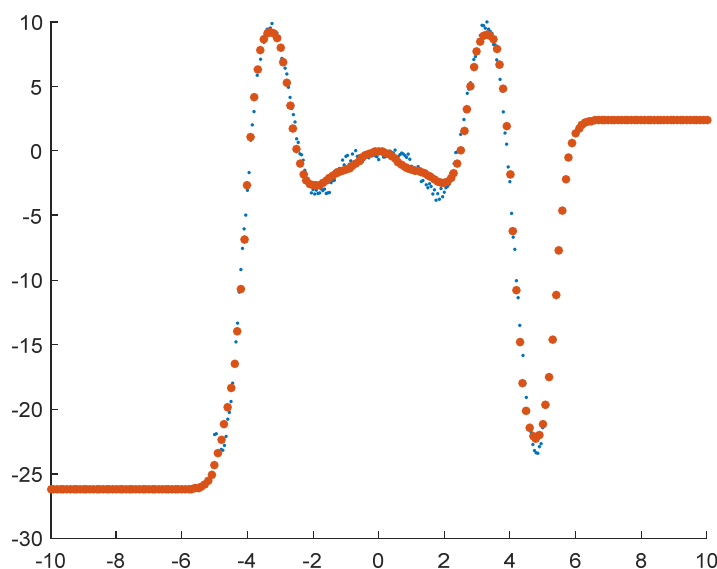


Рис. 3.6. Аппроксимированная функция

Видно, что в диапазоне обучающей выборки предсказание ИНС очень близко к облаку точек исходной функции. Видно также, что в ближайшей окрестности (порядка 10 % диапазона обучающей выборки) ИНС обладает также и некоторой экстраполирующей способностью воспроизводя поведение исходной функции, а в более широкой окрестности сходится к прямой.

Дополнительным инструментом оценки результата обучения ИНС является построение (рис. 3.7) функций регрессии выходных значений ИНС (*Output*) от целевых значений (*Target*), которые были

заданы в обучающей выборке. Данная диаграмма вызывается кнопкой **Regression** (см. рис. 3.4).

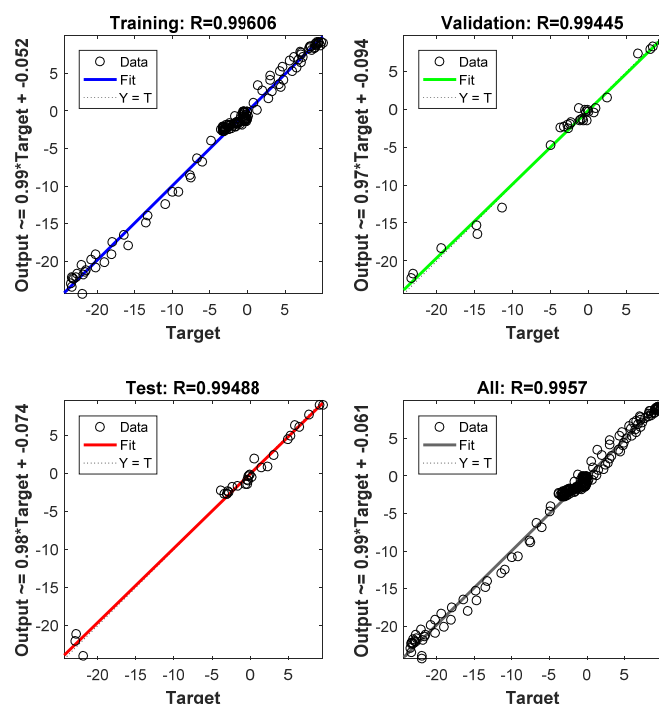


Рис. 3.7. Корреляция расчетных и обучающих выходов

В случае, если построенная ИНС аппроксимирует исходные данные неудовлетворительно, искать решение можно с помощью следующих приемов:

- 1) наращивание структуры сети – числа слоев и числа нейронов;
- 2) наращивание объема обучающей выборки;
- 3) увеличение размерности пространства входных параметров, если таковое имеется;
- 4) замена алгоритма обучения ИНС.

В нашем примере обученность сети Net достаточна.

Порядок выполнения работы

1. Для заданной функции построить ее табличные значения (количество значений должно быть достаточным для того, чтобы аппроксимированная функция визуально совпадала с табличными значениями).

2. Использовать нейронную сеть для аппроксимации этих табличных значений и нахождения аппроксимированной функции. Обучение нейронной сети выполнять с помощью функции train.

3. Вывести на график табличные значения (точками) и аппроксимированную функцию (линией).

4. Найти численное значение погрешности результата аппроксимации, вывести на экран.

Индивидуальные задания

Таблица 3.1

Индивидуальные задания

Вариант	Вид функции	Промежуток нахождения решения	Контрольный вопрос
1	$(1,85-x)*\cos(3,5x-0,5)$	$x \in [-10,10]$	1
2	$\cos(\exp(x))/\sin(\ln(x))$	$x \in [2,4]$	2
3	$\sin(x)/x^2$	$x \in [3.1,20]$	3
4	$\sin(2x)/x^2$	$x \in [-20,-3,1]$	4
5	$\cos(2x)/x^2$	$x \in [-20,-2,3]$	5
6	$(x-1)\cos(3x-15)$	$x \in [-10,10]$	6
7	$\ln(x)\cos(3x-15)$	$x \in [1,10]$	1
8	$\cos(3x-15)/\text{abs}(x)=0$	$x \in [-10,-0.3), (0.3,10]$ $x \in [-0.3,0.3]$	2
9	$\cos(3x-15)*x$	$x \in [-9,6, 9,1]$	3
10	$\sin(x)/(1+\exp(-x))$	$x \in [0,5,10]$	4
11	$\cos(x)/(1+\exp(-x))$	$x \in [0,5,10]$	5
12	$(\exp(x)-\exp(-x))\cos(x)/(\exp(x)+\exp(-x))$	$x \in [-5,5]$	6
13	$(\exp(-x)-\exp(x))\cos(x)/(\exp(x)+\exp(-x))$	$x \in [-5,5]$	1
14	$\cos(x-0,5)/\text{abs}(x)$	$x \in [-10,0),(0,10]$, min	2
15	$\cos(2x)/\text{abs}(x-2)$	$x \in [-10,2),(2,10]$, max	3

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить графическое и табличное представление обучающего множества.
5. Представить программу, написанную в среде MatLab, и результаты аппроксимации функций.
6. Выводы.

Контрольные вопросы

1. Что такое аппроксимация?
2. Синтаксис функции *feedforwardnet*.
3. Параметры обучения при использовании для обучения функции *train*.
4. Виды прекращения обучения сети при использовании функции обучения *train*.
5. Охарактеризуйте способ обучения, приведенный в примере.
6. Способы нахождения погрешности результата.

Лабораторная работа 4 **КЛАССИФИКАЦИЯ С ПОМОЩЬЮ СЕТЕЙ КОХОНЕНА**

Цель работы: изучение модели слоя Кохонена и алгоритма обучения без учителя; создание и исследование модели слоя Кохонена в системе MatLab.

4.1. Классификация без учителя при помощи ИНС

Постановка задачи классификации

При обучении без учителя раскрывается внутренняя структура данных или корреляции между образцами в обучающей выборке. Выходы нейронной сети формируются самостоятельно, а настраиваемые параметры изменяются по алгоритму, учитывающему только входные сигналы и производные от них внутренние переменные системы. Такое обучение называют ещё неуправляемым. В результате такого обучения объекты или примеры обучающей выборки распределяются по классам, причем сами классы и даже их количество могут быть не определены заранее.

Для реализации такого подхода необходимо решить две главные задачи:

1. Разработать методы разбиения входных примеров на классы без учителя – задача обучения;
2. Выработать правила отнесения текущего входного образа к некоторому классу – задача распознавания.

Из общей логики процесса классификации следует, что разбиение какого-либо множества экземпляров на классы должно быть основано на использовании достаточно общих свойств классифицируемых объектов. В один класс должны попасть объекты, схожие между собой с точки зрения исследуемой предметной области. Одним из самых распространенных методов выделения классов является компактность объектов в пространстве описывающих их показателей. Каждому классу в пространстве признаков соответствует обособленная группа точек, каждая из которых описывает один образ объекта классифицируемого множества. Тогда задача классификации сводится к разделению в многомерном пространстве на части множества точек.

Процесс разбиения множества образов объектов на классы в многомерном пространстве называется кластеризацией. Для решения этой задачи необходимо формально задать способ проведения границы между классами. В результате получения новой информации

о классах возможна коррекция существующих правил классификации объектов – дообучение правил классификации.

Сети конкурентного обучения

В процессе развития нейроинформатики выработан специализированный тип ИНС, основанных на конкурентном обучении (competitive learning). Идея конкурентного обучения заключается в том, что нейроны выходного слоя ИНС соревнуются между собой за право активации. При этом активируется, как правило, лишь один из нейронов – победитель в слое ИНС, то есть внутри слоя реализуется принцип «победитель получает все».

Сеть Кохонена

Класс ИНС, именуемых «картами Кохонена» – одна из разновидностей нейронных сетей, использующих обучение без учителя. Идея подобных сетей принадлежит финскому ученому Тойво Кохонену (1982 год). Основной принцип работы карты Кохонена – введение в правило обучения нейронов ИНС информации об их расположении.

Разрабатывая свою идею Тойво Кохонен воспользовался аналогией ИНС со свойствами человеческого мозга, топология коры которого представляет собой свернутый складками плоский лист. В результате такой топологии головного мозга участки, ответственные за близкие части тела, примыкают в нем друг к другу таким образом, что все изображение человеческого тела отображается на двумерную поверхность коры.

При помощи самоорганизующихся карт Кохонена решаются следующие задачи:

1. Моделирование.
2. Прогнозирование.
3. Поиск закономерностей в больших массивах данных.
4. Выявление наборов независимых признаков.
5. Сжатие информации.

Наиболее востребованное на практике применение самоорганизующихся карт Кохонена – решение задачи кластеризации или классификации без учителя.

Существует два важных приложения карт Кохонена в области кластеризации:

1. **Разведочный анализ данных.** Сеть Кохонена способна выделять кластеры в наборах данных, а также вычислять меру близости классов. Эта информация помогает аналитику улучшить свое пони-

мание структуры данных, что важно как для понимания предметной области, так и для уточнения модели ИНС. Если ИНС распознает в имеющихся данных классы, то их можно зафиксировать в правилах классификации, что в дальнейшем позволит нейросети решать задачи классификации. Кроме того, карты Кохонена могут быть использованы и для решения тех задач классификации, где классы predetermined заранее. В таких задачах ИНС может зафиксировать сходство между различными классами.

2. Обнаружение новых явлений. Карта Кохонена распознает кластеры во множестве экспериментальных данных и строит правила их кластеризации. Исследователю наиболее интересны случаи, когда сеть обнаруживает новый класс – набор данных, непохожий ни на один из известных образцов. Такая информация может иметь очень серьезные приложения как в маркетинге, так и в практической политологии (URL: <https://theins.ru/politika/38490>).

Обучение по Хеббу

Рассмотрим базовый для обучения ИНС без учителя алгоритм Хебба.

В таких алгоритмах подстройка синапсов может проводиться только на основании информации доступной в нейроне, т. е. его состояние Y_j и уже имеющихся коэффициентов W . На этих соображениях, а также на аналогии с принципами самоорганизации нервных клеток живых организмов построен алгоритм обучения Хебба (рис. 4.1). В него заложено следующее правило подстройки весов синапсов:

$$W_{ij}(t) = W_{ij}(t-1) + \alpha y_i^{(n-1)} y_j^{(n)}, \quad (4.1)$$

где $y_i^{(n-1)}$ – выходное значение i -го нейрона $(n-1)$ -го слоя; $y_j^{(n)}$ – выходное значение j -го нейрона n -го слоя; $W_{ij}(t), W_{ij}(t-1)$ – весовые коэффициенты синапса, соединяющего i -й и j -й нейроны на итерациях t и $t-1$ соответственно; α – коэффициент скорости обучения.

Анализируя формулу (4.1), можно видеть, что данный алгоритм обучения будет усиливать прежде всего связи между соседними нейронами.

Вариантом данного алгоритма является дифференциальный метод обучения Хебба, в соответствии с которым подстройка весов синапсов производится по формуле

$$W_{ij}(t) = W_{ij}(t-1) + \alpha \left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1) \right] \left[y_j^{(n)}(t) - y_j^{(n)}(t-1) \right]. \quad (4.2)$$

В соответствии с данной формулой сильнее всего будут обучаться веса синапсов, соединяющих нейроны с наиболее динамично изменяющимися выходами.

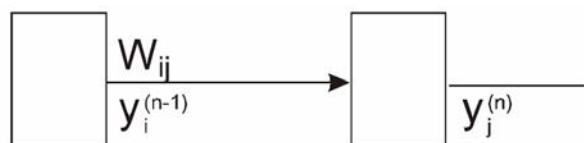


Рис. 4.1. Нейронная сеть
(обозначения см. в формуле (4.1))

Алгоритм обучения Хебба относится к классу алгоритмов без учителя и в полном виде выглядит следующим образом:

1. Стартовая инициализация всех весов синапсов производится небольшими случайными значениями.

2. На входы сети подается входной пример, ИНС срабатывает и передает сигнал возбуждения по всем слоям. При каждом повторе этого шага попеременно предъявляются все примеры обучающей выборки.

3. На основании рассчитанных сетью выходных значений Y_i , Y_j по формулам (4.1) и (4.2) производится настройка синаптических весов.

4. Переход на шаг 2 до тех пор, пока выходные сигналы ИНС не стабилизируются с заданной точностью (разница весов между шагами t и $t+1$ станет достаточно малой).

Реакция ИНС на каждый класс входных примеров не задается заранее и на первых итерациях представляет собой комбинация состояний нейронов, определяющихся случайным распределением весов при инициализации сети. В процессе обучения сеть обобщает похожие образы, все более уверенно относя их к одному классу.

Обучение ИНС по Кохонену

В алгоритме обучения Кохонена заложена подстройка весов синапсов на основании их значений, полученных на предыдущей итерации

$$W_{ij}(t) = W_{ij}(t-1) + \alpha \left[y_j^{(n-1)} - W_{ij}(t-1) \right]. \quad (4.3)$$

В соответствии с формулой (4.3) обучение заключается в минимизации разности между значениями входных сигналов на синапсах нейронов, поступающих с выходов нейронов предыдущего слоя и значениями весов его синапсов.

Структура алгоритма обучения по Кохонену сходна с алгоритмом Хебба, но на шаге 3 из всего слоя выбирается 1 нейрон, значение весов синапсов которого максимально подобны вектору значений входного примера, а затем подстройка весов синапсов этого нейрона выполняется по формуле (4.3). Такой подход реализует принцип «победитель получает все» или, как его еще называют, конкурентное обучение.

Описанная в данном пункте процедура обучения ИНС формирует разбиение множества входных примеров на кластеры, присущие входным данным.

В правильно обученной ИНС для каждого входного примера X активизируется только один выходной нейрон-победитель, соответствующий кластеру, в который попадает текущий входной пример X .

Методы определения нейрона-победителя

Для процедур конкурентного обучения разработано два метода определения нейрона-победителя.

Основой первого из них является скалярное произведение $\bar{X} \cdot \bar{W}$. При таком подходе максимальное значение дает нейрон-победитель. Вектора \bar{X} и \bar{W} как правило не подвергаются нормализации.

Для каждого выходного нейрона производится расчет скалярного произведения

$$y_o = \sum_i W_{io} \cdot x_i = \bar{W}_o^t \cdot \bar{x}, \forall o \neq k, y_o \leq y_k. \quad (4.4)$$

Нейрон, у которого это значение максимально, выбирается победителем.

На следующем шаге для каждого нейрона-победителя устанавливается значение $y_k = 1$, а для остальных: $y_{o \neq k} = 0$.

Синаптические веса нейрона-победителя корректируются следующим образом:

$$\bar{W}_k(t+1) = \frac{\bar{W}_k(t) + \alpha [\bar{x}(t) - \bar{W}_k(t)]}{\left\| \bar{W}_k + \alpha [\bar{x}(t) - \bar{W}_k(t)] \right\|}. \quad (4.5)$$

В формуле (4.5) деление на норму $\left\| \bar{W}_k + \alpha [\bar{x}(t) - \bar{W}_k(t)] \right\|$ позволяет сохранять значения векторов \bar{W} нормализованными.

В результате расчета по формуле (4.5) вектор синаптических весов смещается («подкручивается») в сторону вектора входного примера X (рис. 4.2).

Процесс обучения заключается в том, что сети по очереди предъявляется текущий пример обучающей выборки \bar{X}_i , определяется ближайший к нему вектор синаптических весов нейрона-победителя, который «подкручивается» по направлению к \bar{X}_i . В результате многократного повторения такой процедуры вектора весов нейронов вращаются по направлению к тем областям пространства входных параметров, где находится много примеров обучающей выборки. Эти области как раз и соответствуют кластерам (рис. 4.3).

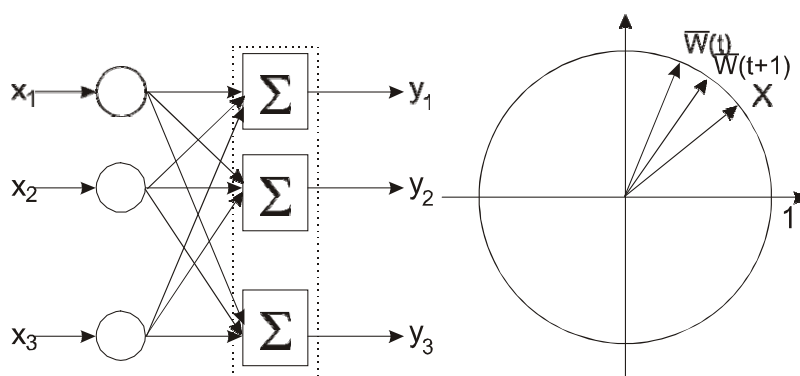


Рис. 4.2. Вращение весового вектора в процессе обучения

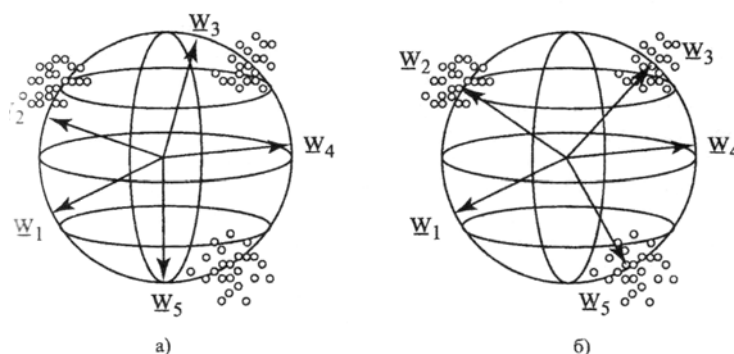


Рис. 4.3. Установление векторов весов синапсов в центры кластеров.

Альтернативным методом обучения сети Кохонена является использование евклидова расстояния в пространстве синаптических весов. В этом варианте в качестве победителя выбирается k -й нейрон для которого евклидово расстояние между входным сигналом \bar{X}_i и вектором синаптических весов удовлетворяет условию (рис. 4.4):

$$K : \|\bar{W}_k - \bar{X}_i\| \leq \|\bar{W}_o - \bar{X}_i\|, \forall o \quad (4.6)$$

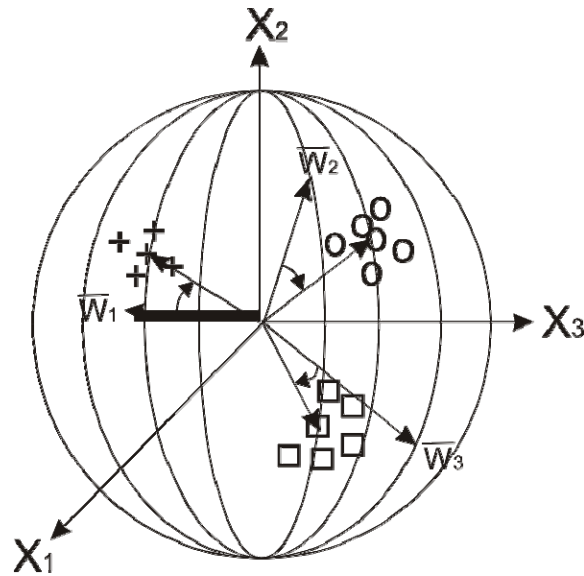


Рис. 4.4. Установление весовых векторов нейронов в центры кластеров

Когда нейрон-победитель определен, его синаптические веса корректируются по подобной формуле

$$\bar{W}_k(t) = \bar{W}_k(t-1) + \alpha(\bar{x}(t) - \bar{W}_k(t)). \quad (4.7)$$

Можно видеть, что в случае, если вектора \bar{X} и \bar{W} нормализованы, то результаты расчетов по формулам (4.4) и (4.6) не отличаются.

4.2. Пример решения задачи классификации сетью Кохонена

Как было показано выше, самоорганизующаяся карта Кохонена потребуется нам, если мы не знаем «правильных», целевых значений выхода сети (значение T). В получившихся в процессе работы графиках мы увидим, что система автоматически определяет и разносит центры кластеров.

Подготовка данных для кластеризации

Зададим z – базу знаний для работы с самоорганизующейся картой Кохонена. Ее размерность – 2×200 . На рис. 4.5 показан код MatLab, формирующий экспериментальные данные. Сначала задается множество значений кластера № 1. Центром кластера будет являться точка (3;0) с небольшим разбросом от центра *rand*(1,30) и количеством отсчетов 30, т. е. $x1$ и $y1$ будет являться массивом размерность 1×30 .

Таким же образом задаётся множество значений для второго кластера с центром $(-3;0)$, третий и четвертый кластеры с центрами $(0;3)$ и $(0;-3)$.

После задания всей базы знаний визуализируем все кластеры на одном графике.

```
% задаем кластер №1
y1=0+rand(1,30);
x1=3+rand(1,30);
% задаем кластер №2
y2=0+rand(1,30);
x2=-3+rand(1,30);
% задаем кластер №3
y3=3+rand(1,30);
x3=0+rand(1,30);
% задаем кластер №4
x4=0+rand(1,30);
y4=-3+rand(1,30);
% Отображаем данные на figure (1)
figure (1);
hold on;
plot(x3,y3,'og');
plot(x4,y4,'oy');
plot(x1,y1,'ob');
plot(x2,y2,'or');
grid on
hold off
```

Рис. 4.5. Генерация данных для сети Кохонена

Формирование базы знаний

Предполагается, что целевых значений (T) для примеров обучающей выборки мы не знаем, поэтому заполним их произвольными значениями. Кроме того, перед тем, как подавать это на вход нейросети, соединим всю базу знаний для нейронной сети – координаты примеров x и y в одну матрицу – переменную z . Код MatLab для этой задачи – на рис. 4.6.

Создание и обучение карты Кохонена

Поскольку у нас всего 4 класса и появление новых не ожидается, то достаточно будет задать гексагональную сетку самообучающейся карты Кохонена размерностью 2×2 . Соответствующий код MatLab приведен на рис. 4.7.

```

% Зададим цели наших четырех кластеров
T3(1:30)=10;
T4(1:30)=20;
T1(1:30)=30;
T2(1:30)=40;
% Соединим все цели кластеров
T(1:30)=T1;
T(31:60)=T2;
T(61:90)=T3;
T(91:120)=T4;
% Соединим всю базу знаний в переменной z
x(1:30)=x1;
x(31:60)=x2;
x(61:90)=x3;
x(91:120)=x4;
y(1:30)=y1;
y(31:60)=y2;
y(61:90)=y3;
y(91:120)=y4;
z(1,1:120)=x;
z(2,1:120)=y;

```

Рис. 4.6. Формирование базы знаний для карты Кохонена

```

% Зададим самообучающуюся карту
net = selforgmap([2 2]);
% Проведем цикл обучения
net = train(net, z);

```

Рис. 4.7. Код MatLab на создание и обучение карты Кохонена

Так же, как и в других типах ИНС, в карте Кохонена в MatLab имеются параметры для тонкой настройки этой сети, которые можно посмотреть при обучении сети, такие как количество эпох, градиент, погрешность и т. п. В командном окне к ним нужно обращаться как имя_сети.команда, например net.time (рис. 4.8).

На рис. 4.9 показано, как выглядит карта Кохонена, получившаяся после генерации и обучения (код на рис. 4.7).

На рис. 4.10 можно видеть, как распределились по кластерам примеры, которые мы использовали при обучении.

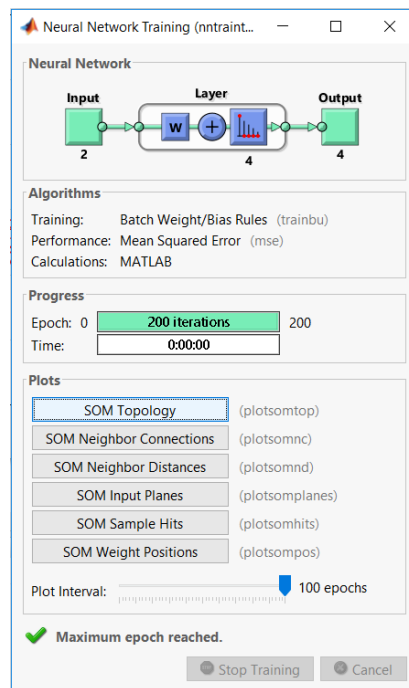


Рис. 4.8. Окно обучения карты Кохонена в MatLab

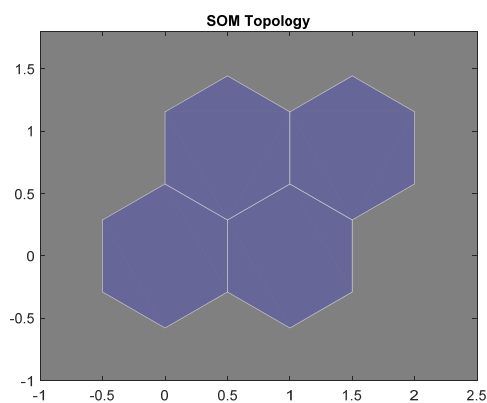


Рис. 4.9. Топология самоорганизующейся карты Кохонена

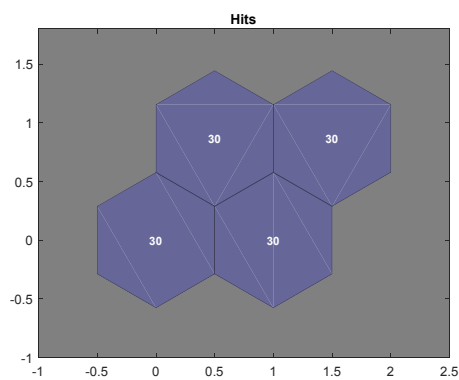


Рис. 4.10. Распределение примеров по кластерам

На рис. 4.11 показано, как распределились в результате обучения центры кластеров нейронной сети Кохонена.

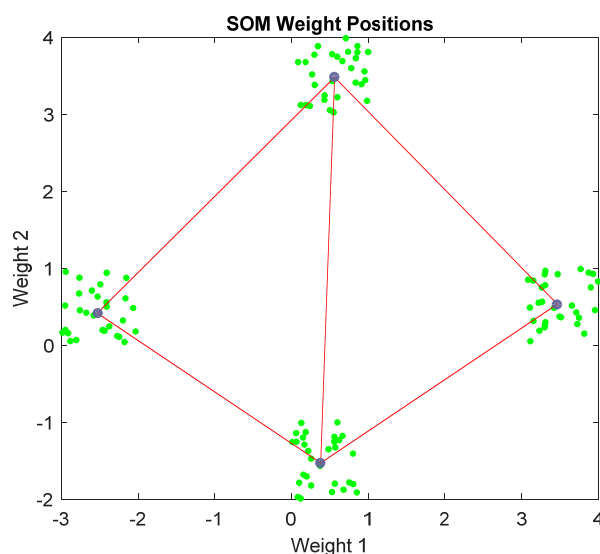


Рис. 4.11. Центры кластеров нейронной сети Кохонена

Тестирование сети Кохонена

Проверим получившуюся сеть на тех примерах, с помощью, которых была обучена ИНС.

`a = sim(net,z)`

Аналогично можно проверить на других примерах.

Порядок выполнения работы

1. Построить нейронную сеть, которая производит классификацию на основе слоя Кохонена с помощью функции ***newsc***. Произвести начальную инициализацию весов нейронной сети с помощью функции ***init***.

2. Построить обучающую выборку, которая позволяет правильно классифицировать заданную проверочную выборку. Координаты точек обучающей выборки должны быть подобраны геометрически. Для этого первоначально нанести точки проверочной выборки на график, ориентировочно построить вокруг каждой из этих точек несколько точек обучающей выборки, затем записать координаты этих точек в массив обучающей выборки. На каждую из точек проверочной выборки должно приходиться не менее 3 точек обучающей выборки.

3. Показать обучающую выборку на графике с помощью функции **plot**. На графике должны быть координатные оси, подписи по осям и название самого графика.

4. Создать самоорганизующуюся карту Кохонена при помощи команды **selforgmap**.

5. Произвести обучение нейронной сети на составленной обучающей выборке с использованием функции **train**.

6. Используя обученную нейронную сеть, произвести классификацию проверочного множества, с помощью функции **sim**. Показать результат классификации в командном окне MatLab и на графике.

Индивидуальные задания

Таблица 4.1

Задания для индивидуальной работы

№ варианта	Кол-во классов для классификации	Координаты точек проверочного множества, номер класса, к которому принадлежит данная точка	Контрольный вопрос
1	3	[0;0]–1; [1;1]–2; [–1;–1]–3;	1
2	4	[2;1]–1; [1;0]–2; [0;–1]–3;	2
3	5	[0;0]–1; [1;1]–1; [–1;–1]–3;	3
4	6	[1;0]–1; [–1;1]–2; [–1;–1]–2;	4
5	7	[0;0]–1; [1;1]–2; [–1;–1]–3;	5
6	8	[0;0]–1; [1;1]–1; [–1;–1]–1;	6
7	3	[–1;0]–1; [–1;1]–2; [–1;–1]–3;	1
8	4	[0;1]–1; [1;–1]–2; [–1;–1]–3;	2
9	5	[2;0]–1; [1;1]–2; [–1;1]–2;	3
10	6	[0;0]–1; [1;1]–2; [–1;–1]–1;	4
11	7	[0;0]–1; [1;1]–1; [–1;–1]–2;	5
12	8	[0;0]–1; [1;1]–2; [–1;–1]–3;	6
13	3	[0;0]–1; [1;1]–1; [–1;–1]–1;	1
14	4	[0;1]–1; [1;–1]–2; [–1;–1]–3;	2

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить графическое и табличное представление обучающего множества.
5. Представить графическое и табличное представление проверочных точек.
6. Представить программу, написанную в среде MatLab, и результаты классификации.
7. Выводы.

Контрольные вопросы

1. Архитектура ИНС Кохонена.
2. Правило нахождения количества нейронов в сети для распознавания заданного числа классов.
3. Алгоритм обучения.
4. Анализ информации, выдаваемой функцией *display*.
5. Результат, который возвращает функция *sim*.
6. Параметры функций *selforgmap*.

Лабораторная работа 5

НЕЙРОСЕТЕВОЕ ПРОГНОЗИРОВАНИЕ ВРЕМЕННЫХ РЯДОВ

Цель работы: изучение способа прогнозирования временных ряда с помощью ИНС в системе MatLab.

5.1. Понятие прогнозирования временных рядов

Временной ряд (ВР) – это последовательность значений, описывающих протекающий во времени процесс, измеренных в последовательные моменты времени, обычно через равные временные интервалы, т. е. ряд наблюдений $x(t_1), x(t_2), \dots, x(t_N)$ анализируемой величины $x(t)$, произведенных в последовательные моменты времени t_1, t_2, \dots, t_N .

Данные типа временных рядов широко распространены в самых разных областях человеческой деятельности. В экономике это ежедневные цены на акции, курсы валют, еженедельные и месячные объемы продаж, годовые объемы производства и т.п.

Варианты трендов во временных рядах приведены на рис. 5.1: линейный тренд (рис. 5.1, а), случайные колебания (рис. 5.1, б), сложный цикл (рис. 5.1, в).

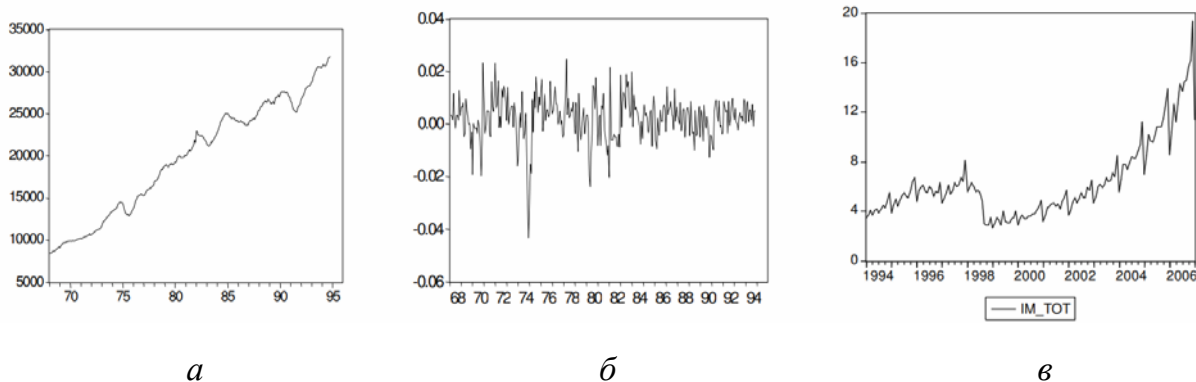


Рис. 5.1. Тренды ВР
(обозначения см. в тексте)

Цели анализа ВР:

- краткое описание характерных особенностей ряда;
- подбор статистической модели, описывающей ВР;
- предсказание будущих значений на основе прошлых наблюдений;
- управление процессом, порождающим ВР.

Сформулируем постановку задачи одношагового прогнозирования:

Имеется временной ряд Y , заданный значениями измерений величины, в предшествующие моменты времени $Y_{t-k}, Y_{t-k+1}, \dots, Y_t$ где t – текущий момент времени; k – глубина исторической выборки а также временным шагом (или «лагом» от английского *lag* – запаздывание).

Необходимо спрогнозировать величину измеряемого параметра в следующий момент времени, на основе анализа прошлых значений, а также истории изменения других факторов, влияющих на динамику прогнозируемой величины.

Существует два различающихся подхода к прогнозированию. При первом важным является минимальная величина ошибки прогнозирования. При втором (который, в частности, применяется при прогнозе биржевых курсов для управления торгами) важно правильное определение характера дальнейшего изменения значения прогнозируемой величины – рост или уменьшение.

При решении задачи прогнозирования временного ряда необходимо найти значение (или несколько величин, вектор) величины прогнозируемого параметра Y в будущий момент времени на основе известных предыдущих значений $\bar{Y}, \bar{V}, \bar{C} : Y_{t+1} = g(\bar{Y}, \bar{V}, \bar{C})$:

$$Y_{t+1} = g(Y_{t-k} \dots Y_t, V_{1,t-k} \dots V_{1,t}, V_{n,t-k} \dots V_{n,t}, C_{1,t-k} \dots C_{1,t}, C_{n,t-k} \dots C_{n,t}). \quad (5.1)$$

Формула (5.1) описывает основное соотношение, определяющее прогноз временного ряда.

5.2. Определение структуры нейронной сети для прогноза ВР

Определение структуры нейронной сети предполагает задание ее архитектуры (число слоев, число нейронов на каждом слое), а также функции активации нейронов.

При использовании многослойного персептрона для решения задач аппроксимации функций (в том числе и задач прогнозирования) нет необходимости в числе скрытых слоев больше одного. Таким образом, предлагается использовать нейронную сеть, имеющую в своем составе входной, один скрытый и выходной слою. Число нейронов первого (то есть входного) слоя $n^{(1)}$ зависит от количества вход-

ных факторов и глубины ретроспективной выборки по каждому фактору и определяется по формуле

$$n^{(1)} = \sum P \cdot k_p, \quad (5.2)$$

где P – число входных факторов; k_p – глубина ретроспективной выборки по p -му фактору.

Таким образом, число нейронов входного слоя равно числу элементов входных данных.

Выходной слой содержит один нейрон, выход которого представляет собой полученное в результате работы нейронной сети значение прогноза. При необходимости получения одновременно нескольких выходных значений (например, котировок продажи и покупки ценной бумаги) предпочтительнее не увеличивать число выходных нейронов, а обучать несколько отдельных нейронных сетей, каждая из которых имеет один выход. В пользу предлагаемого подхода свидетельствуют результаты исследований. Так, в частности, в результате экспериментов было установлено, что точность прогноза нейросети с двумя выходами хуже, чем двух нейросетей с единственным выходом при обучении на одних и тех же данных.

Число нейронов второго (скрытого) уровня первоначально предлагается принимать равным полусумме числа нейронов входного и выходного слоев:

$$n^{(2)} = \frac{\sum P \cdot k_p + 1}{2}. \quad (5.3)$$

В процессе определения и настройки параметров нейронной сети число скрытых нейронов может варьироваться с целью увеличения точности прогноза. При этом порядок их количества можно приблизительно оценить с помощью теоремы Колмогорова и следствия из нее. В соответствии с этими теоретическими результатами непрерывную функцию можно аппроксимировать с любой точностью при помощи трехслойной нейронной сети, которая имеет N входных, $2N + 1$ скрытых и один выходной нейрон. Таким образом, число $2N + 1$ теоретически можно считать верхней границей для числа нейронов на скрытых уровнях. Однако данная оценка справедлива для наилучшей сети, соответствующей глобальному минимуму адаптивного рельефа ошибки в пространстве весов. На практике достичь глобального минимума обычно не удастся. Кроме того, имея $2N + 1$ скрытых нейронов, сеть способна в точности запомнить обучающие примеры. В то же время в реальности от нее требуется их обобщить, выявив скрытые

закономерности. Таким образом, оценка Колмогорова по сути сделана для другой задачи и в данном случае служит лишь ориентиром. В ходе экспериментов эмпирическим путем было установлено, что число скрытых нейронов может составлять от $N/2$ до $3N$.

В качестве функции активации нейронов предлагается использовать биполярную сигмоиду. Сигмоидная функция традиционно используется в сетях типа «многослойный персептрон» так как она является гладкой, дифференцируемой, а ее производная выражается через саму функцию, что важно для обучения сети. В тоже время рациональная сигмоида с областью определения $[0;1]$ имеет недостаток, связанный с ее несимметричностью относительно оси абсцисс, который уменьшает скорость обучения. Поэтому в данной лабораторной работе предлагается применять биполярную сигмоидную функцию. Эксперименты показали, что область определения биполярной функции не оказывает существенного влияния на процесс обучения. Для определенности в дальнейшем будет использоваться функция с областью определения $[-1;1]$.

5.3. Подготовка обучающей выборки для прогнозирования ВР

Стандартным приемом подготовки данных для обучения нейронной сети необходимо сформировать из массива данных обучающие примеры. Каждый из обучающих примеров должен содержать вход и требуемый (целевой) выход нейросети, который сеть должна научиться выдавать в ответ на данный вход. Выборка входных данных должна содержать историю предыдущих значений по факторам, используемым при построении прогноза. Целевой выход формируется из будущих значений прогнозируемой величины, которые известны на этапе обучения. В качестве иллюстрации ниже приведен фрагмент реальных данных, и на их основе сформирован обучающий пример (табл. 5.1, 5.2).

Таблица 5.1

Исходные данные

Название фактора	Значение нормированной котировки на дату					Требуемый выход сети
	04.06.97	05.06.97	06.06.97	09.06.97	10.06.97	
US Treas-5	0	-1,552	0,789	-0,052	0,29	
Dow Jones			-1,304	-0,393	-,058	
ОГВВЗ-5	-0,124	-0,124	-0,358	-0,245	0	0,462

Таблица 5.2

Вид сформированного примера

0	-1,552	0,789	-0,052	0,29	-1,304	-0,393	-0,058	-0,124	-0,124	-0,358	-0,245	0	0,462
---	--------	-------	--------	------	--------	--------	--------	--------	--------	--------	--------	---	-------

Как видно, в обучающем примере исходные данные вытягиваются в строку вида <вход нейросети><выход нейросети> (в данном случае первые пять значений – US Treasuries 5, затем три значения – Dow Jones, потом пять значений ОГВВЗ 5 и завершает строку значение требуемого выхода сети (т. е. будущая котировка ОГВВЗ 5) (см. табл. 5.2).

Количество нейронов входного слоя сети соответствует числу элементов входных данных, содержащихся в обучающем примере, а число нейронов выходного слоя – соответственно числу эталонных выходов (в данном случае – один).

Совокупность обучающих примеров создается из массива данных с информацией о значениях в предшествующие моменты времени с помощью метода «скользящего окна». Для формирования обучающего примера на массив исходных данных накладывается временное «окно». При этом данные о значениях входных факторов, попавшие в него, формируют обучающий пример. Следующий пример формируется путем сдвига «окна» на один временной интервал вперед. В качестве ширины «окна» по каждому фактору используется ранее определенная глубина ретроспективной выборки.

Сформированные примеры разделяются на два подмножества: обучающее и контрольное. Первое подмножество используется для непосредственного обучения сети, второе – для обучения не используется и служит для оценки способности сети работать с незнакомыми данными. Обычно число примеров обучающего множества относится к числу примеров контрольного множества как 2:1 или 3:1.

5.4. Построение программы в среде MatLab, использующей нейронные сети для прогнозирования временных рядов

Для построения программы, решающей задачу прогнозирования, будем считать, что временной ряд изменяется по определенному закону (представленному таблицей вариантов), на который влияют также случайные помехи.

Получение исходного временного ряда

Построим модель временного ряда $z(t) = a \cdot t + b \cdot \sin(w \cdot t) + c \cdot \text{randn}(t)$, содержащую тренд, циклическую и случайную компо-

ненту. Количество значений ряда примите равным 150. Значения a , b , w , и c выберите произвольно (рис. 5.2).

```
t=1:100;  
L=length(t);  
z=10*sin(0.2*t)+0.2*t+0.3*randn(1,L);  
x=[0 z(1:L-1);0 0 z(1:L-2)];
```

Рис. 5.2. Код MatLab для подготовки данных ВР

Создание и обучение нейросети

С помощью функции *feedforwardnet* создадим нейронную сеть с двумя нейронами в основном слое и одним – в выходном, а затем обучим её, используя предшествующие значения ряда с помощью функции *train*. С помощью функции *sim* запустим нейронную сеть и получим значения прогноза. Затем построим графики для ряда $z(t)$ и прогноза $Y(t)$ в одном окне, сравним расхождения прогноза и последних 50 значений ряда $z(t)$. Соответствующий код приведен на рис. 5.3.

```
net = feedforwardnet([2 1]);  
net.trainParam.epochs = 100;  
net = train(net,x,z);  
y = sim(net,x);  
y(101)=sim(net,[z(100) z(99)]');  
y(102)=sim(net,[y(101) z(100)]');  
for i=1:50  
y(102+i)=sim(net,[y(101+i), y(100+i)]' );  
end  
plot(z, 'b'), hold on, grid on  
plot(y, 'r'), hold off
```

Рис. 5.3. Код MatLab для обучения сети на прогноз ВР

Прогноз временного ряда можно видеть на рис. 5.4. Видно, что обученная ИНС достаточно точно прослеживает тенденцию временного ряда еще на протяжении пятидесяти точек.

В качестве упражнения выполните следующие действия:

1. Вычислите и постройте график ошибки $h(t) = |z(t) - y(t)|$.
2. Увеличьте количество нейронов в основном слое и повторите п. 3–5.
3. Используйте 5 предшествующих состояний, задайте 2 нейрона в основном слое и повторите п. 3–5.

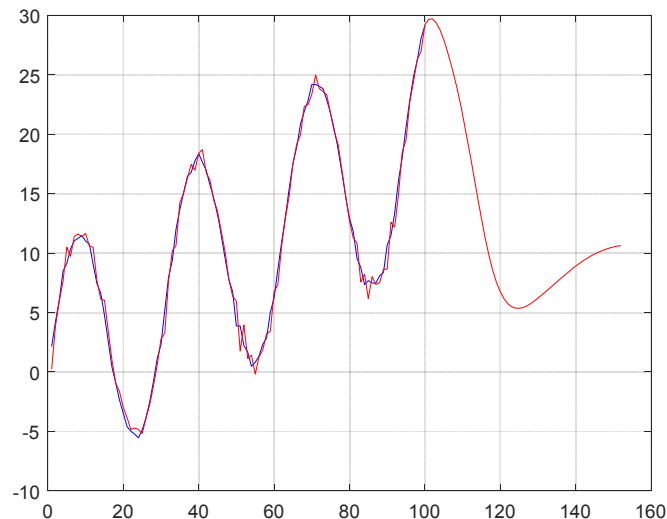


Рис. 5.4. Прогноз ВР нейросетью

Порядок выполнения работы

1. Построить временной ряд, который представляет собой функцию. Вид функции указан в вариантах задания.

2. К полученному временному ряду добавить шум в размере максимум 20 % от амплитуды сигнала.

3. Построить выборку для обучения. Для этого на основании временного ряда строится 5 рядов с задержкой от 1 до 5. Для построения ряда с задержкой 5 берутся от 1 до $n-5$ элементы выборки, с задержкой 4 берутся от 2 до $n-4$ элементы выборки, с задержкой 3 берутся от 3 до $n-3$ элементы выборки, с задержкой 2 берутся от 4 до $n-2$ элементы выборки, с задержкой 1 берутся от 5 до $n-1$ элементы выборки. Здесь n – длина исходного временного ряда.

4. Построить проверочную выборку. Поскольку длина выборки для обучения на 5 элементов меньше длины исходного временного ряда, для построения взять элементы от 6 до n -го.

5. Временной ряд разбить на 2 части: использующиеся для обучения сети и для проверки. Размеры массивов должны относиться друг к другу приблизительно как 3:1.

6. Построить нейронную сеть для прогнозирования. Число слоев – 2. Активационная функция первого слоя – гиперболический тангенс, второго – линейная. Число нейронов первого слоя взять достаточным для удовлетворительного прогнозирования (10–100), число нейронов второго слоя – 1.

7. Произвести обучение сети на обучающем множестве, привести график исходного ряда и спрогнозированного, а также погрешности прогнозирования.

8. Произвести проверку работы сети на проверочном множестве, привести график исходного ряда и спрогнозированного, а также погрешности прогнозирования.

Индивидуальные задания

Таблица 5.3

Вид временного ряда и диапазон его изменения

Вариант	Вид функции	Промежуток нахождения решения	Контрольные вопросы
1	$(1,85-t) \cdot \cos(3,5t-0,5)$	$t \in [0,10]$	1
2	$\cos(\exp(t))$	$t \in [2,4]$	2
3	$\sin(t) \cdot t$	$t \in [3.1,20]$	3
4	$\sin(2t) \cdot t$	$t \in [3.1,20]$	4
5	$\cos(2t) \cdot t$	$t \in [2.3,20]$	5
6	$(t-1)\cos(3t-15)$	$t \in [0,10]$	6
7	$\cos(3t-15)$	$t \in [1,10]$	7
8	$\cos(3t-15)/\text{abs}(t)$	$t \in [0,0.3),(0.3,10]$	8
9	$\cos(3t-15) \cdot t$	$t \in [0.9,1]$	9
10	$(\exp(t)-\exp(-t))\cos(t)/(\exp(t)+\exp(-t))$	$t \in [0,5]$	1
11	$(\exp(-t)-\exp(t))\cos(t)/(\exp(t)+\exp(-t))$	$t \in [0,5]$	2
12	$\cos(t-0,5) \cdot \text{abs}(t)$	$t \in (0,10]$	3
13	$\cos(2t) \cdot \text{abs}(t-2)$	$t \in (2,10]$	4
14	$0,5 \cdot t + \sin(2 \cdot t)$	$t \in [0,10]$	5
15	$2 \cdot t + \cos(0,5 \cdot t)$	$t \in [0,10]$	6

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить графическое и табличное представление обучающего множества.

5. Представить графическое и табличное представление проверочных точек.
6. Представить программу, написанную в среде MatLab, и результаты прогнозирования временных рядов.
7. Выводы.

Контрольные вопросы

1. Что такое временной ряд?
2. Опишите метод прогнозирования значений временного ряда с помощью нейронной сети.
3. В чём заключается процесс обучений нейронной сети?
4. Как количество предшествующих значений, взятых для обучения влияет на качество прогнозирования?
5. Как количество нейронов влияет на качество прогнозирования?
6. С помощью какой НС выполняется прогнозирование временного ряда?
7. Перечислите принципы формирования обучающей выборки.
8. Назовите принципы формирования проверочной выборки.

Лабораторная работа 6

СЕТИ ХОПФИЛДА

Цель работы: в работе исследуются свойства нейронной сети Хопфилда, которая рассматривается как модель ассоциативной памяти, позволяющей восстановить объект по ограниченному набору зашумленных признаков. Необходимо изучить динамику переходного процесса в сети Хопфилда.

6.1. Теоретические аспекты сетей Хопфилда

Формальное описание сети Хопфилда

ИНС Хопфилда состоит из N искусственных нейронов, выходной сигнал каждого нейрона связан с входными сигналами остальных нейронов, образуя обратную связь. Архитектура сети изображена на рис. 6.1.

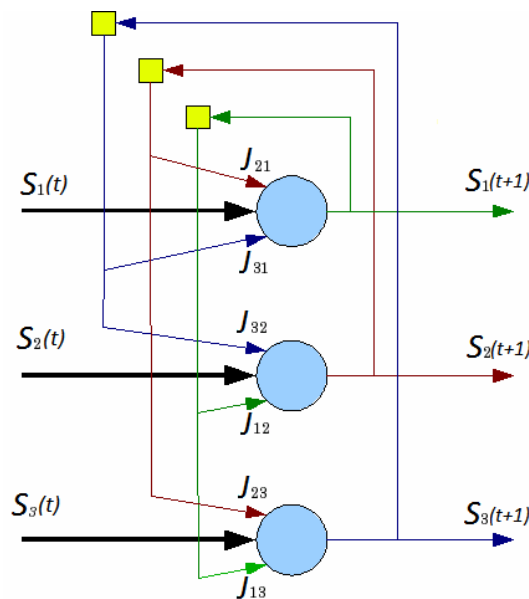


Рис. 6.1. Архитектура ИНС Хопфилда

Каждый нейрон может находиться в одном из 2-х состояний:

$$S(t) \in \{-1; +1\}, \quad (6.1)$$

где $S(t)$ состояние нейрона в момент t . «Возбуждению» нейрона соответствует «+1», а «торможению» – «-1». Дискретность состояний нейрона отражает нелинейный, пороговый характер его функционирования и известный в нейрофизиологии как принцип «все или ничего».

Динамика состояния во времени i -го нейрона в сети из N нейронов описывается дискретной динамической системой:

$$S_i(t+1) = \text{sign} \left[\sum_{j=1}^N J_{i,j} S_j(t) \right], \quad i, j \in 1, \dots, N, \quad (6.2)$$

где $J_{i,j}$ – матрица весовых коэффициентов, описывающих взаимодействие выходов i -го нейрона со входами j -го нейрона.

Следует обратить внимание, что случаи $J_{i,j} = 0$ и $\sum_{j=1}^N J_{i,j} S_j(t) = 0$ не рассматриваются.

Обучение ИНС Хопфилда и их устойчивость к шуму

Обучение ИНС Хопфилда выходным сигналам (образам) ζ_μ^{in} сводится к вычислению значений элементов матрицы $J_{i,j}$.

Процесс обучения можно формально описать следующим образом:

Пусть необходимо обучить ИНС распознавать M образов, обозначенных $\{\zeta_\mu^{in}, \mu = 1, \dots, M\}$. Входной образ $\bar{\zeta}_\mu^{in}$ представляет собой: $\bar{\zeta}_\mu^{in} = \zeta_\mu^{in} + \zeta'$ где ζ' – шум, наложенный на исходный образ ζ_μ^{in} .

Фактически, обучение нейронной сети – определение нормы в пространстве образов $\|\zeta_\mu^{in} - \bar{\zeta}_\mu^{in}\|$. Исходя из этого решение ИНС Хопфилда задачу очистки входного образа от шума можно описать как минимизацию этого выражения.

Важной характеристикой ИНС является отношение числа ключевых образов M , которые могут быть запомнены сетью, к N – числу ее нейронов: $\alpha = \frac{M}{N}$. Для сети Хопфилда значение α не превышает значения 0,14, т. е. необходимо около 7 нейронов для того, чтобы запомнить один ключевой образ.

Вычисление квадратной матрицы размера $M \times M$ для ключевых образов производится по правилу Хебба:

$$J_{i,j} = \frac{1}{N} \sum_{\mu=1}^M [\zeta_{i,\mu}^{in} \cdot \zeta_{j,\mu}^{in}], \quad (6.3)$$

где $\zeta_{i,\mu}^{in}$ – означает j -й элемент образа ζ_μ^{in} .

Рассмотрев формулу (6.3) можно видеть, что, в силу коммутативности операции умножения, равенство $J_{i,j} = J_{j,i}$ будет всегда выполняться.

Предъявляемый для распознавания входной образ, соответствующий начальным данным для системы и служащий начальным условием для динамической системы (6.2):

$$S_i = \bar{\zeta}_{\mu}^{in}. \quad (6.4)$$

Уравнений (6.1), (6.2), (6.3), (6.4) достаточно для определения ИНС Хопфилда. Далее рассмотрим ее реализацию в системе MatLab.

6.2. Реализация ИНС Хопфилда

Создание сети Хопфилда

Рассмотрим ИНС Хопфилда, имеющую два нейрона и два устойчивых состояния, отображаемых векторами $[1 \ -1]$ и $[-1 \ 1]$ (рис. 6.2).

```
T=[+1 -1;-1 +1];
plot(T(1,:),T(2,:),'r*');
axis([-1.1 1.1 -1.1 1.1]);
title('Пространство векторов сети Хопфилда');
xlabel('a(1)');
ylabel('a(2)');
```

Рис. 6.2. Подготовка данных для ИНС Хопфилда

Визуализация исходных данных на рис. 6.3.

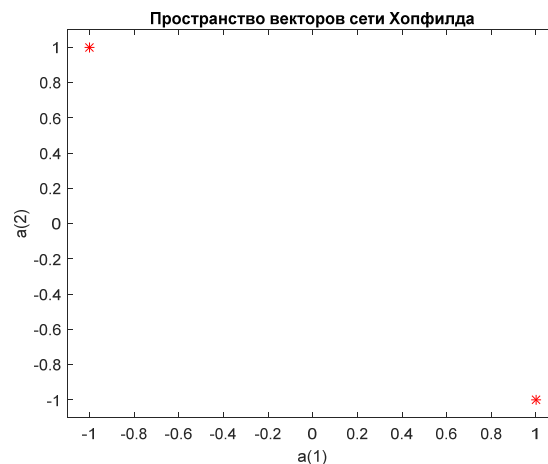


Рис. 6.3. Исходные вектора ИНС Хопфилда

Создание ИНС Хопфилда

Создадим ИНС Хопфилда с именем *Net* и проверим ее работу, подав на вход векторы, соответствующие устойчивым точкам (рис. 6.4).

```
Net=newhop(T); % Создание НС Хопфилда
[Y,Pf,Af] = sim(Net,2,[], T);
Y % Опрос сети Хопфилда
```

Рис. 6.4. Создание и опрос ИНС Хопфилда

Проверим результат опроса, убедимся, что ИНС работает корректно. Если сеть работает правильно, она должна выдавать эти же векторы без каких-либо изменений (см. рис. 6.6).

Решение задач при помощи ИНС Хопфилда

Теперь подадим на ее вход произвольный вектор (рис. 6.5).

```
a={rands(2,1)};% Задание случайного вектора a =[2x1] типа double
[y,Pf,Af]=sim(Net,{1 50},{},a);
plot(T(1,:),T(2,:),'r*');
axis([-1.1 1.1 -1.1 1.1]);
record = [cell2mat(a) cell2mat(y)];
start = cell2mat(a);
hold on;
plot (start (1,1), start (2,1),' bx', record(1, :), record(2, :));
xlabel('a(1)'); ylabel('a(2)');
title('Результат работы ИНС Хопфилда');
```

Рис. 6.5. Тестирование ИНС Хопфилда

На рис. 6.6 представлены векторы $[1 \ -1]$ и $[-1 \ 1]$, отражающие устойчивые состояния НС, а на рис. 6.7 виден результат работы ИНС в случае подачи произвольного вектора.

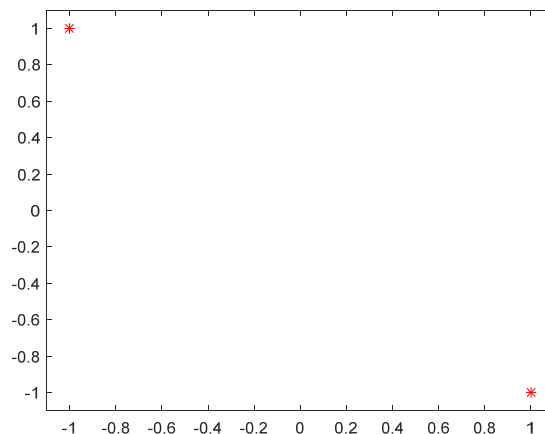


Рис. 6.6. Устойчивые точки ИНС Хопфилда

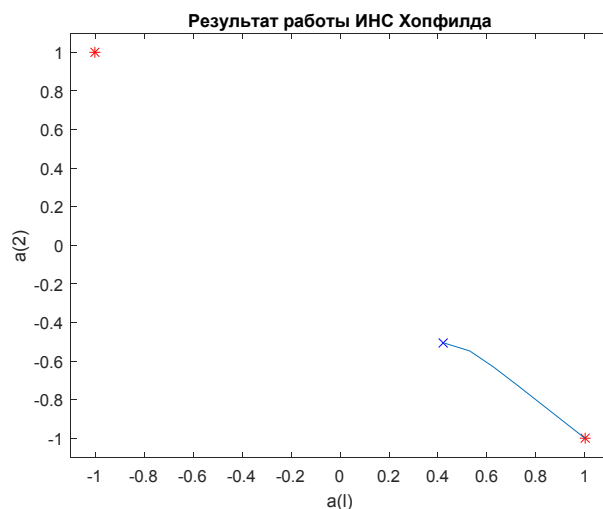


Рис. 6.7. Иллюстрация работы ИНС Хопфилда

Порядок выполнения работы

1. Подготовьте данные для ИНС Хопфилда заданного размера.
2. Создайте сеть при помощи команды *newhop*;
3. Протестируйте ИНС Хопфилда на случайных данных. Подготовьте отчет по результатам ее функционирования

Варианты индивидуальных заданий

Таблица 6.1.

Исходные данные для создания сети Хопфилда

Вариант	Размерность входного пространства ИНС Хопфилда	Контрольные вопросы
1	3	1
2	4	2
3	5	3
4	3	4
5	4	5
6	5	6
7	3	7
8	4	1
9	5	2
10	3	3
11	4	4
12	5	5
13	3	6
14	4	7
15	5	1

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить графическое и табличное представление исходных точек.
5. Представить программу, написанную в среде MatLab, и результаты решения зашумленных данных сетью Хопфилда.
6. Выводы.

Контрольные вопросы и задания

1. Расскажите о топологии сети Хопфилда.
2. Расскажите об обучении сети Хопфилда.
3. Опишите процесс воспроизведения информации в сети Хопфилда.
4. Поясните зависимость максимального количества образов, запоминаемых сетью, от ее размера.
5. Перечислите варианты использования сети Хопфилда.
6. Какие функции используются для создания ИНС Хопфилда?
7. Как проводится опрос сети Хопфилда? Можно ли на основании результатов опроса НС сделать вывод о правильной работе сети?

Лабораторная работа 7 **НЕЙРОСЕТЕВОЕ РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ**

Цель работы: исследование возможностей распознавания изображений (на примере печатных символов) с помощью ИНС.

7.1. Задача распознавания образов

Задача распознавания образов заключается в классификации предъявляемых объектов на несколько (классов) категорий. Образами в этой задаче называются объектами.

Решение задачи классификации основывается на прецедентах.

Прецедент – ранее классифицированный (то есть для него известна правильная классификация) объект, принимаемый в качестве образца для решения классификационных задач.

Следует отметить, что в современном естественно-научном мировоззрении идея принятия решений на основе анализа прецедентов основополагающая.

Для решения задачи распознавания будем считать, что все объекты или явления разбиты на конечное число классов. Для каждого класса известно и изучено конечное число объектов – прецедентов.

Тогда задача распознавания образов заключается в отнесении ранее не встречавшихся распознаваемый объектов к какому-либо из заданных классов.

Задача распознавания образов играет очень важную роль во многих классах систем искусственного интеллекта. Рассмотрим некоторые классы.

1. Системы компьютерного зрения. Обеспечивают получение изображений через цифровые камеры и их расшифровку в символьном виде, описание того, какие объекты присутствуют на изображении, каковы их координаты и параметры движения, как они расположены относительно друг друга и т. д.

2. Символьное распознавание – оцифровка букв или цифр:

- Optical Character Recognition (OCR);
- ввод и хранение документов;
- компьютерное письмо (Pen Computer);
- обработка банковских чеков;
- обработка бумажной почты.

3. Медицинская диагностика:
 - маммография, рентгенография;
 - постановка диагноза по истории болезни;
 - электрокардиограмма.
4. Дефектоскопия на производстве.

7.2. Решение задачи распознавания символов

Подготовка эталонных образов

Примером набора эталонных образов является последовательность из десяти цифр от 0 до 9. Число образов в этом примере $N = 10$. В случае, когда каждый класс образов характеризуется только одним эталоном, имеем число классов, также равное N .

Каждый эталонный образ формируется в виде графического файла в битовом формате. Расширение файла определяется используемыми в среде MatLab типами графических файлов. Рекомендуемый тип файла – *tif*.

Для создания графических файлов образов можно использовать любой графический редактор, это может быть «Paint» или «Adobe Photoshop».

При создании каждого файла образа необходимо проделать следующую последовательность операций:

1. Создать новый файл, задав его параметры:
 - имя (в формате, допустимом операционной системой);
 - ширина ($N1$ пикселей);
 - высота ($N2$ пикселей);
 - цветовой режим.
2. Используя инструменты графического редактора создать требуемый образ символа.
3. Сохранить созданный образ в виде файла типа *tif* в рабочей папке.

На рис. 7.1 приведены примеры графических символов цифр при $N1 = 10$, $N2 = 12$ пикселей.



Рис. 7.1. Примеры графических символов цифр

Для успешного обучения ИНС недостаточно формирования лишь одного обучающего образа для каждого класса (типа символа) образов. Это связано с тем, что распознаваемые образы (на этапе работы ИНС в режиме распознавания) чаще всего отличаются от обучающих по целому ряду причин:

- различие шрифтов и стилей печатных символов;
- погрешности оцифровки;
- низкое качество исходного изображения т. д.

Для обучения ИНС подготовим в каталоге программы набор файлов, содержащих *dataset* – зашумленные изображения. Программа, готовящая *dataset* приведена на рис. 7.2.

```
CharsetNames=['0_char.tif';'1_char.tif';'2_char.tif';'3_char.tif';...
'4_char.tif';'5_char.tif';'6_char.tif';'7_char.tif';'8_char.tif';...
'9_char.tif'];
for i=1:size(CharsetNames)
A=CharsetNames(i,:);
X=imread(A);
for j=1:50
B=rand(size(X));
C=X+((B>0.7));
NewName=strcat(int2str(j),'_',A)
imwrite(C,NewName);
end
end
```

Рис. 7.2. Программа создания зашумленных примеров

Вероятность зашумления в одном пикселе определяется строчкой $C = X + ((B > 0,7))$. При линейно распределенной вероятности шума чем выше число в скобке, тем ниже вероятность шума. При показателе 0,7 зашумленные символы обучающей выборки выглядят так, как на рис. 7.3.



Рис. 7.3. Зашумленные обучающие примеры

Считывание графических файлов в обучающую выборку

Эталонный образ каждого символа представлен в виде вектора-столбца $[N]$, число элементов N которого равно числу признаков (иначе говоря, N – размерность пространства признаков). Такой вектор-столбец формируется из двумерного массива-изображения $[N1, N2]$, который, в свою очередь, формируется при считывании графического файла образа с помощью команд *imread* (процедура чтения графического файла) и *reshape* (процедура преобразования двумерного массива в одномерный вектор-столбец). Контрастность изображения обеспечивает использование векторов и матриц типа *logical*.

Соответствующий фрагмент кода приведен на рис. 7.4.

```
% Чтение исходных данных из графических файлов
CharsetNames=['0_char.tif';'1_char.tif';'2_char.tif';'3_char.tif';...
'4_char.tif';'5_char.tif';'6_char.tif';'7_char.tif';'8_char.tif';...
'9_char.tif'];
k=0;
N1=size(CharsetNames);
N2=N1(1);
for i=1:N2
    A=CharsetNames(i,:);
    for j=1:20
        k=k+1;
        ReadName=strcat(int2str(j),'_',A);
        X=logical(imread(ReadName));
        L=size(X);
        N=L(1)*L(2);
        XR(k,:)=reshape(X,[N,1]);
        Y0=zeros(N,1);
        Y0(i)=1;
        YR(k,:)=logical(Y0');
    end
end
end
```

Рис. 7.4. Код MatLab считывания обучающих образов

Создание и обучение ИНС в среде MatLab

Подготовка к обучению ИНС, включает следующие шаги:

1. Формирование двумерного массива обучающих образов XR , каждый столбец которого представляет собой набор N признаков одного образа, а число столбцов K равно числу обучающих образов.

2. Формирование двумерного массива целевых откликов YR .
3. Создание ИНС с требуемой структурой и параметрами при помощи команды *feedforwardnet*.

4. Запуск процесса обучения ИНС запускается командой ***train***.

Для решения задач распознавания печатных символов будем использовать трехслойную ИНС с один скрытым слоем с числом нейронов равным 50. Выходной слой содержит число нейронов, соответствующее числу классов классификации. Соответствующий код MatLab приведен на рис. 7.5.

```
InData=double(XR');
OutData=double(YR');
nHidden=50;
Net = feedforwardnet(nHidden);
Net.performFcn = 'sse'; % sse квадратичная ошибка
Net.trainParam.goal = 1e-3; % целевое значение ошибки.
Net.trainParam.epochs = 5000; % максимальное число эпох обучения
Net.trainParam.mc = 0.95; % Momentum constant.
Net.trainParam.lr = 0.0001;
Net.layers{1}.transferFcn = 'tansig';
Net.layers{2}.transferFcn = 'logsig';
Net.trainFcn = 'traingda';
Net=init(Net);
Net.IW{1,1}=Net.IW{1,1}+1e-2*randn;
Net.b{1}=Net.b{1}+1e-3*randn;
Net.LW{2,1}=Net.LW{2,1}+1e-2*randn;
Net.b{2}=Net.b{2}+1e-2*randn;
% обучаем сеть в заданных настройках
Net = train(Net,XR',YR');
```

Рис. 7.5. Код MatLab создания и обучения ИНС

Распознавание печатных символов обученной ИНС

Работа ИНС *Net*, заключающаяся в формировании отклика Y при входном воздействии в виде вектора-столбца $X[N]$ производится командой *sim(Net,X)*.

Тестирование работы ИНС при распознавании печатных символов с различной степенью искажения производится с помощью кода MatLab, приведенного на рис. 7.6.

```

% тестируем на зашумленных образах
for i=1:size(CharsetNames)
for j=1:10
A=CharsetNames(i,:);
X=logical(imread(A));
B=rand(size(X));
C=X+((B>0.7));
L=size(X);
N=L(1)*L(2);
XTest=double(reshape(X,[N,1]));
YTest=sim(Net,XTest)'
end
end

```

Рис. 7.6. Код MatLab создания и обучения ИНС

ИНС демонстрирует хорошее распознавание даже при сильном искажении. Для объективной оценки качества работы ИНС необходимо вычисление вероятностных характеристик распознавания.

Порядок выполнения работы

1. Подготовить графические файлы эталонных образов для символов, заданных преподавателем.
2. В среде MatLab создать и обучить ИНС, предназначенную для распознавания печатных символов.
3. Исследовать зависимость качества работы ИНС:
 - от степени искажения символов;
 - числа нейронов в скрытом слое;
 - функций активации нейронов;
 - алгоритмов обучения ИНС.

Качество работы ИН следует оценить вероятностями правильной классификации $P_{\text{пр}}(i)$ образа i -го класса, $i = 1, \dots, N$. Оценка вероятностей $P_{\text{пр}}(i)$ производится по формуле

$$\hat{P}_{\text{пр}}(i) = \frac{N_{\text{пр}}}{N_0},$$

где $N_{\text{пр}}$ – число правильных распознаваний образа i -го класса; N_0 – общее число распознаваний образов i -го класса. Число $N_{\text{пр}}$ определяется экспериментально при запуске программы при значениях $N_0 = 10 \dots 100$.

Варианты индивидуальных заданий

Таблица 7.1

Набор символов и их размеры

Вариант	Набор символов	Размеры изображений	Контрольные вопросы
1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	8×12	1
2	A, B, C, D, E, F, G, H, I, J	10×15	2
3	K, L, N, O, P, Q, R, S, T, U	12×18	3
4	Q, R, S, T, U, V, W, X, Y, Z	8×12	4
5	a, b, c, d, e, f, g, h, i, j	10×15	5
6	k, l, n, o, p, q, r, s, t, u	12×18	6
7	q, r, s, t, u, v, w, x, y, z	8×12	7
8	А, Б, В, Г, Д, Е, Ё, Ж, З, И	10×15	1
9	Й, К, Л, М, Н, О, П, Р, С, Т	12×18	2
10	У, Ф, Х, Ц, Ч, Ш, Щ, Ъ, Ь, Э	8×12	3
11	Х, Ц, Ч, Ш, Щ, Ъ, Ь, Э, Ю, Я	10×15	4
12	а, б, в, г, д, е, ё, ж, з, и	12×18	5
13	й, к, л, м, н, о, п, р, с, т	8×12	6
14	у, ф, х, ц, ч, ш, щ, ъ, ь, э	10×15	7
15	х, ц, ч, ш, щ, ъ, ь, э, ю, я	12×18	1

Содержание отчета

1. Титульный лист.
2. Задание, учитывая свой вариант.
3. Теоретические сведения.
4. Представить изображения исходных символов и пример генерации зашумленных.
5. Представить программу, написанную в среде MatLab, и результаты распознавания зашумленных и исходных символов.
6. Выводы.

Контрольные вопросы

1. Что называется образом и классом в задаче распознавания образов?
2. Чем определяется эффективность системы распознавания?
3. Какие функции работы с изображениями в MatLab вы знаете?
4. Каковы возможности использования нейронных сетей для решения задач распознавания?
5. В чём заключается процесс обучений нейронной сети?
6. Как количество нейронов влияет на качество распознавания?
7. Как влияет на качество распознавания зашумленность обучающей выборки?

ЗАКЛЮЧЕНИЕ

Искусственные нейронные сети (ИНС) в современном мире применяются для широкого диапазона задач, простирающегося от управления боем до продажи товаров в супермаркетах. Потенциальными приложениями ИНС являются те области, где человеческий интеллект малоэффективен, а обычные вычисления трудоемки или недостаточно адекватны. Этот диапазон приложений как минимум не уступает тому, что обслуживается традиционными алгоритмами, и следует ожидать, что ИНС займут свое место наряду с обычными вычислениями в качестве дополнения не меньшего объема и важности.

Происходящее в середине – конце 10-х годов XXI века возрождение интереса к нейроинформатике вызвано целым рядом как теоретических, так и прикладных достижений. Появление алгоритмов глубокого обучения и сверточных ИНС, а также рост доступных вычислительных мощностей открыли возможности использования вычислений в сферах, до этого относящихся лишь к области человеческого интеллекта. Так, в 2018 году ИНС начали лучше человека распознавать изображение и голос. Появились возможности создания машин, способность которых учиться и запоминать все более напоминает мыслительные процессы высшей нервной деятельности.

Разработано и разрабатывается множество обучающих алгоритмов, каждый из которых имеет свои сильные и слабые стороны, предназначен для решения своих классов задач. Все еще не до конца решена задача, чему в принципе может обучиться ИНС и далеко не для всех приложений понятно, как должно проводиться обучение.

Представленные в пособии типовые архитектуры ИНС и нейросетевые парадигмы дают представление об искусстве конструирования сетей в целом. Многие из более сложных архитектур являются лишь модификациями тех, что были рассмотрены. Понимание и владение описанными архитектурами и парадигм позволит обучающимся в дальнейшем осознанно следить за прогрессом в такой быстро развивающейся области, как нейроинформатика.

Для желающих продолжать саморазвитие в области нейроинформатики в списке литературы приводятся ссылки на более полные и точные источники. Важным направлением дальнейшего развития в области изучения ИНС будет освоение глубокого обучения (Deep Learning) и сверточных нейронных сетей (Convolutional Network).

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сайт [Электронный ресурс]. – Режим доступа: <http://matlab.exponenta.ru/neuralnetwork/>. – Загл. с экрана.
2. Сайт [Электронный ресурс]. – Режим доступа: <https://www.mathworks.com/campaigns/offers/machine-learning-with-matlab.html>. – Загл. с экрана.
3. Ануфриев, И. Е. MatLab 7. Наиболее полное руководство в подлиннике / И. Е. Ануфриев, А. Б. Смирнов, Е. Н. Смирнова. – СПб. : БХВ-Петербург, 2005. – 1104 с.
4. Горбань, А. Н. Нейронные сети на персональном компьютере / А. Н. Горбань, Д. А. Россиев. – Новосибирск : Наука. Сиб. отд-ние, 1996. – 276 с.
5. Комарцова, Л. Г. Нейрокомпьютеры : учеб. пособие для вузов / Л. Г. Комарцова, А. В. Максимов. – 2-е изд. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2004. – 400 с.
6. Медведев, В. С. Нейронные сети. MatLab 6 / В. С. Медведев, В. Г. Потемкин. – М. : Диалог-МИФИ, 2002. – 496 с.
7. Уоссермен, Ф. Нейрокомпьютерная техника: теория и практика / Ф. Уоссермен. – М. : Мир, 1992. – 118 с.
8. Ясницкий, Л. Н. Искусственный интеллект. Элективный курс : учеб. пособие / Л. Н. Ясницкий. – М. : БИНОМ, Лаборатория знаний, 2011. – 200 с.

Учебно-практическое издание

Доррер Михаил Георгиевич

МОДЕЛИРОВАНИЕ НЕЙРОННЫХ СЕТЕЙ В СИСТЕМЕ MATLAB

Лабораторный практикум

Редактор *П. С. Бороздов*

Оригинал-макет и верстка *М. А. Светлаковой*

Подписано в печать 15.06.2021. Формат 60×84/16. Бумага офисная.

Печать плоская. Усл. печ. л. 5,7. Уч.-изд. л. 6,7. Тираж 50 экз.

Заказ . С 136/21.

Санитарно-эпидемиологическое заключение

№ 24.49.04.953.П.000032.01.03 от 29.01.2003 г.

Редакционно-издательский отдел СибГУ им. М. Ф. Решетнева.

660037, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31.

E-mail: rio@mail.sibsau.ru. Тел. (391) 201-50-99.

Отпечатано в редакционно-издательском центре

СибГУ им. М. Ф. Решетнева.

660049, г. Красноярск, просп. Мира, 82. Тел. (391) 227-69-90.