

ОГЛАВЛЕНИЕ

Список основных сокращений.....	6
Введение	7
Глава 1. Методические рекомендации по организации лабораторного практикума.....	10
§ 1.1. Рекомендуемые темы лабораторных работ.....	10
§ 1.2. Методические указания по проведению лабораторных работ	10
§ 1.3. Структура и порядок оформления отчета	12
§ 1.4. Порядок защиты лабораторных работ.....	13
§ 1.5. Рекомендуемая литература для подготовки к лабораторным работам и защиты отчетов по ним.....	13
Глава 2. Лабораторная работа «Моделирование персептрона в среде MATLAB»	15
§ 2.1. Цель работы	15
§ 2.2. Нейрон персептрона.....	15
2.2.1. Простой нейрон.....	15
2.2.2. Нейрон с векторным входом	16
2.2.3. Функции активации	17
2.2.4. Модель нейрона	22
§ 2.3. Персептрон.....	23
2.3.1. Архитектура сети	23
2.3.2. Моделирование персептрона средствами MATLAB.....	24
2.3.3. Инициализация параметров.....	30
§ 2.4. Задание для лабораторной работы.....	32
§ 2.5. Структура отчета	33
Глава 3. Лабораторная работа «Линейные нейронные сети. Обучение линейной сети в MATLAB»	35
§ 3.1. Цель работы	35
§ 3.2. Линейные сети	35

3.2.1. Структура линейной сети	35
3.2.2. Архитектура сети	37
3.2.3. Создание линейной сети	38
3.2.4. Конструирование линейной сети	39
§ 3.3. Процедуры настройки линейной сети	42
3.3.1. Процедура настройки линейной сети с использованием функции <code>adapt</code>	42
3.3.2. Процедура настройки линейной сети с использованием функции <code>newlind</code>	47
§ 3.4. Обучение линейной сети. Обучающее правило наименьших квадратов.....	50
§ 3.5. Задание для лабораторной работы.....	59
§ 3.6. Структура отчета	62
Глава 4. Лабораторная работа «Моделирование нейронных сетей в MATLAB».....	63
§ 4.1. Цель работы	63
§ 4.2. Пакет Neural Network Toolbox.....	63
§ 4.3. Работа с нейронной сетью в командном режиме	67
§ 4.4. Использование GUI-интерфейса пакета нейронных сетей.....	74
4.4.1. Создание нейронной сети.....	75
4.4.2. Обучение нейронной сети	79
4.4.3. Работа с созданной сетью.....	82
§ 4.5. Моделирование нейронных сетей при помощи Simulink	86
4.5.1. Средства Simulink для работы с нейронными сетями	86
4.5.2. Обзорщик разделов библиотеки Simulink	87
4.5.3. Создание модели	88
4.5.4. Окно модели.....	90
4.5.5. Выполнение моделирования	93
4.5.6. Разработка НС для решения систем обыкновенных дифференциальных уравнений в системе Simulink	96

§ 4.6. Задание для лабораторной работы.....	104
§ 4.7. Структура отчета	105
Глава 5. Лабораторная работа «Моделирование и реализация нейро-нечеткой сети в среде MATLAB»	106
§ 5.1. Цель работы	106
§ 5.2. Нечеткая сеть TSK.....	106
§ 5.3. Гибридная сеть как адаптивная система нейро-нечеткого вывода	109
§ 5.4. Моделирование и реализация нейро-нечеткой сети в среде MATLAB.....	110
5.4.1. Описание ANFIS-редактора.....	111
5.4.2. Синтез нейро-нечеткой сети в среде MATLAB	118
§ 5.5. Задание для лабораторной работы.....	126
§ 5.6. Структура отчета	127
Приложение. Список функций Neural Network Toolbox	129
Список литературы	138

СПИСОК ОСНОВНЫХ СОКРАЩЕНИЙ

ИИ — искусственный интеллект;
ИНС — искусственная нейронная сеть;
НМ — нечеткое множество;
НС — нейронная сеть;
ОДР — область допустимых решений;
ОДУ — обыкновенные дифференциальные уравнения;
ОУ — объект управления;
ПК — персональный компьютер;
ПЭВМ — персональная электронно-вычислительная машина;
ТНМ — теория нечетких множеств;
УУ — управляющее устройство;
ФИ — функция истинности;
ФП — функция принадлежности.

ANFIS — Adaptive Neuro-Fuzzy Inference System (адаптивная нейро-нечеткая система);
FIS — Fuzzy Inference System (нечеткая система вывода);
FLT — Fuzzy Logic Toolbox (пакет нечеткой логики);
GUI — Graphic User Interface (ГИП — графический интерфейс пользователя);
NNT — Neural Network Toolbox (пакет нейронные сети);
TSK-сеть — нейронная сеть (Takagi — Sugeno — Kang'a — Такаги — Сугено — Канга).

ВВЕДЕНИЕ

Термин «искусственный интеллект» используется для обозначения большого направления научных и прикладных исследований. Наука под названием «искусственный интеллект» входит в комплекс наук, в рамках которых проводятся исследования в области информационных и компьютерных технологий [1–3].

Сейчас, в XXI в., имеется достаточно большое количество успешно действующих систем ИИ в различных сферах человеческой деятельности (военное дело, исследование космоса, океана, медицина, геология и т. п.). Эти системы позволяют провести комплексную автоматизацию производственных процессов, проводить разнообразные исследования в реальном масштабе времени, управлять машинами без участия человека и т. п. Методы искусственного интеллекта позволили создать эффективные компьютерные программы в самых разнообразных, ранее считавшихся недоступными для формализации и алгоритмизации сферах человеческой деятельности, таких как медицина, биология, зоология, социология, культурология, политология, экономика, бизнес, криминалистика и т. п. Идеи обучения и самообучения компьютерных программ, накопления знаний, приемы обработки нечетких и неконкретных знаний позволили создать программы, творящие чудеса [1–5].

Одно из направлений ИИ — это нейронные сети (НС) и нейрокомпьютеры, в основе которого лежит идея создания искусственных интеллектуальных устройств по образу и подобию человеческого мозга. Нейронные системы основаны на моделировании функций высшей нервной системы человека. Это направление получило исключительное развитие в XXI в. Одной из ключевых технологий нейронных сетей является обучение НС на примерах [1, 4, 6, 7].

Целями освоения дисциплины «Системы искусственного интеллекта» являются [4, 6–10]:

- ознакомление студентов с вопросами автоматизации обработки информации с использованием нечетких систем, сведениями об основных системах искусственного интеллекта;

- формирование у студентов знаний методологических основ искусственного интеллекта, направлений исследований в области ИИ, методов и моделей ИИ, базовых положений теории нечетких множеств;

- приобретение умений и практических навыков в решении слабоформализуемых задач.

Задачи дисциплины [4, 6–10]:

- приобретение необходимых знаний, умений и навыков в моделировании интеллектуальной деятельности человека;

- изучение основных методологических положений теории ИИ, системологических принципов построения систем ИИ, методов и моделей ИИ, основных положений теории нечетких множеств;

- освоение способов технической реализации мыслительной деятельности человека техническими средствами;

- приобретение практических навыков применения методов и моделей ИИ, основных положений теории нечетких множеств для решения слабоформализуемых задач.

Целями предлагаемого лабораторного практикума являются закрепление знаний, приобретенных на лекциях, анализ полученных результатов и их применение для решения практических задач [6, 7].

При проведении лабораторных работ, представленных в данном пособии, студенты приобретают

знания:

- в применении основных методов теории нечетких множеств для решения задач принятия решений, методов ИИ для построения систем ИИ, в компьютерном моделировании систем ИИ для решения слабоформализуемых задач;

- в изучении назначения и особенностей системы MATLAB;

- в изучении модели нейрона и принципов построения на основе нейрона персептрона, архитектуры линейной нейронной сети и процедуры настройки параметров линейных нейронных сетей посредством прямого расчета в системе MATLAB;

- в изучении средств и методов MATLAB, пакетов Neural Network Toolbox и Simulink для моделирования и исследования нейронных сетей;

- в изучении методов моделирования и принципов функционирования нейро-нечетких сетей с использованием средств и методов MATLAB;

умения и навыки:

- в освоении базовых приемов моделирования персептрона в среде MATLAB;

- в создании и исследовании моделей линейных нейронных сетей, в решении задач с помощью линейной нейронной сети в системе MATLAB;

- в освоении базовых приемов моделирования и исследования нейронных сетей в среде MATLAB и в применении нейронных сетей для аппроксимации функций;

- по конструированию нейро-нечетких сетей в среде MATLAB;

- в анализе полученных результатов.

Лабораторный практикум по дисциплине выполняется с применением ЭВМ с использованием системы MATLAB (MatLab) и интегрированного пакета Microsoft Office for Windows (версии XP и более поздние версии). При выполнении лабораторного практикума могут применяться стандартные пакеты программ (типа системы Turbo Delfi, EXCEL, LabView) [6, 7, 11–15].

По результатам работы все студенты составляют индивидуальные отчеты, сдача которых происходит на следующих лабораторных работах по дисциплине.

В учебном пособии приводятся материалы по лабораторным работам, посвященным моделированию персептрона, нейронных и нейро-нечетких сетей в системе MATLAB.

Глава 1

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ ЛАБОРАТОРНОГО ПРАКТИКУМА

§ 1.1. Рекомендуемые темы лабораторных работ

Для направления подготовки «Информатика и вычислительная техника», профиль «Программное обеспечение вычислительной техники и автоматизированных систем» и для направления подготовки «Управление в технических системах», профиль «Управление и информатика в технических системах» предлагаются следующие лабораторные работы.

№ п/п	Тема занятий	Форма занятий	Кол-во часов
1	Моделирование персептрона в среде MATLAB	Лабораторная работа на ПК	4 (4*)
2	Линейные нейронные сети. Обучение линейной сети в MATLAB	Лабораторная работа на ПК	4 (4*)
3	Моделирование нейронных сетей в MATLAB	Лабораторная работа на ПК	4
4	Моделирование и реализация нейронной сети в среде MATLAB	Лабораторная работа на ПК	4

* Для очно-заочной и заочной форм обучения.

§ 1.2. Методические указания по проведению лабораторных работ

Для выполнения лабораторных работ необходимо располагать компьютерным классом с ПК, объединенными в локальную сеть с выходом в INTERNET со средой MATLAB.

При выполнении лабораторных работ второй части лабораторного практикума студенты должны приобрести следующие знания, умения и навыки:

- в изучении модели нейрона и принципов построения на основе нейрона персептрона, архитектуры линейной нейронной сети и процедуры настройки параметров линейных нейронных сетей посредством прямого расчета в системе MATLAB;
- в моделировании и обучении персептронов и линейных нейронных сетей;
- в освоении базовых приемов моделирования персептрона в среде MATLAB;
- в создании и исследовании моделей линейных нейронных сетей, в решении задач с помощью линейной нейронной сети в системе MATLAB;
- в анализе и обобщении полученных результатов [6, 7, 10].

К каждому занятию студенты должны самостоятельно изучить соответствующие разделы теоретического курса, подробно ознакомиться с содержанием и методикой проведения предстоящей работы. С порядком проведения лабораторных работ студенты знакомятся на основе методических указаний на каждую лабораторную работу и указаний, приведенных в данном учебном пособии. При этом студент в своей тетради должен начертить схему проведения исследований, выписать формулы для расчетов и заготовить форму отчета. Рекомендуется все эти данные подготовить в электронном виде с использованием интегрированного пакета Microsoft Office for Windows (версии XP и более поздние версии), включающего в себя приложения: WORD — текстовый процессор; EXCEL — табличный редактор.

В начале каждого занятия в форме краткого опроса проверяется подготовленность к выполнению лабораторной работы, знание мер безопасности. При проведении занятий в специализированном компьютерном классе с применением компьютерных программ допущенные к работе студенты приступают к работе на ПК на основе методических указаний по лабораторной работе [6, 7].

После выполнения лабораторной работы студенты представляют результаты расчетов на проверку преподавателю для определения правильности их выполнения. При проведении лабораторных работ

учебная группа делится на подгруппы численностью не более 12 человек.

§ 1.3. Структура и порядок оформления отчета

По материалам работы каждым студентом составляется отчет по установленной форме. Отчет должен быть оформлен аккуратно, с четким подробным заполнением всех разделов бланка и в полном соответствии с требованиями ГОСТ по оформлению текстовых документов.

Отчет по лабораторной работе должен содержать:

- 1) исходные данные по каждому заданию в виде таблиц, созданных в табличном процессоре Microsoft EXCEL (на базе другого программного пакета);
- 2) анализ полученных результатов исследований и выводы по работе.

Рекомендуется выполнение рисунков, диаграмм с использованием ПЭВМ. Графики строятся на координатных осях с указанием масштаба и откладываемых физических величин. При построении на одной системе координат нескольких графиков (кривых) их точки отмечаются различными значками. Каждый график должен иметь название и лаконичный текст, поясняющий его целевое назначение, параметрическую зависимость и характерные особенности.

Особое внимание при оформлении отчета обучаемые должны обратить на составление выводов по выполненной работе. В выводах нужно сопоставить результаты проведенных исследований с известными из теоретического курса закономерностями и указать причины наблюдаемых отклонений. Полностью оформленный отчет представляется каждым обучаемым преподавателю на следующей лабораторной работе или в другое установленное преподавателем время, но не ранее чем через неделю после проведения работы.

Представленные в отчете результаты исследований, их порядок получения, расчетов и обработки обучаемые обязаны уметь четко пояснить.

§ 1.4. Порядок защиты лабораторных работ

Цель защиты лабораторной работы заключается в оценке не только практических навыков и умений студентов, но и в проверке их теоретических знаний по данному разделу дисциплины, в оценке их умения анализировать результаты исследований и делать на их основе правильные выводы.

Для подготовки к защите студенты должны ознакомиться с вопросами по данной работе, самостоятельно найти на них ответы, оформить отчет о работе. Вопросы для подготовки к защите лабораторной работы приводятся в методических указаниях по проведению работы и в данном учебном пособии.

Защита, как правило, осуществляется на следующей лабораторной работе. За проведенную работу, оформленный отчет и по результатам опроса преподаватель выставляет дифференцированную оценку с указанием замечаний.

§ 1.5. Рекомендуемая литература для подготовки к лабораторным работам и защиты отчетов по ним

Основная литература

1. Романов, П. С. Системы искусственного интеллекта : учебное пособие / П. С. Романов, И. П. Романова ; под общ. ред. П. С. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — 244 с.

2. Романов, П. С. Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 1 : Моделирование нечетких систем в системе MATLAB. — 115 с.

3. Романов, П. С. Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 2 : Моделирование персептрона и линейных нейронных сетей в системе MATLAB. — 76 с.

4. Романов, П. С. Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 3 : Моделирование нейронных сетей в системе MATLAB. — 115 с.

Дополнительная литература

1. Тимохин, А. Н. Моделирование систем управления с применением MatLab : учебное пособие / А. Н. Тимохин, Ю. Д. Румянцев. — М. : НИЦ ИНФРА-М, 2016. — 256 с.

2. Ясницкий, Л. Н. Искусственный интеллект : учебное пособие. — М. : Бином, 2011. — 200 с.

3. Джонс, Т. М. Программирование искусственного интеллекта в приложениях : пер. с англ. А. И. Осипов. — М. : ДМК Пресс, 2011. — 312 с.

4. Дьяконов, В. П. MATLAB 7.*/R2006/R2007: самоучитель. — М. : ДМК Пресс, 2008. — 768 с.

5. Поршнев, С. В. MATLAB 7. Основы работы и программирования : учебник. — М. : Бином-Пресс, 2011. — 320 с.

6. Леоненков, А. В. Нечеткое моделирование в среде MATLAB и fuzzyTECH. — СПб. : БХВ-Петербург, 2005. — 736 с.

7. Романова, И. П. Системы искусственного интеллекта (контрольная работа) : учебное пособие / И. П. Романова, П. С. Романов ; под общ. ред. П. С. Романова. — Коломна : КИ(ф) МГОУ, 2015. — 38 с.

Глава 2

ЛАБОРАТОРНАЯ РАБОТА «МОДЕЛИРОВАНИЕ ПЕРСЕПТРОНА В СРЕДЕ MATLAB»

§ 2.1. Цель работы

Лабораторная работа выполняется на основе теоретических положений искусственного интеллекта на ПК с использованием систем MatLab (MATLAB) и Microsoft WORD в среде Windows 8/10.

Цели работы:

1) изучение основного элемента нейронной сети — нейрона — и принципов построения на основе нейрона простейшей нейронной сети — перцептрона;

2) получение умений и навыков:

- в освоении базовых приемов моделирования перцептрона в среде MATLAB;
- в анализе полученных результатов.

§ 2.2. Нейрон перцептрона

Рассмотрим основной элемент нейронной сети — нейрон — и построение перцептрона на основе теоретических положений [6, 11–15].

2.2.1. Простой нейрон

Элементарной ячейкой нейронной сети является нейрон. Структура нейрона с единственным скалярным входом показана на рисунке 2.1.

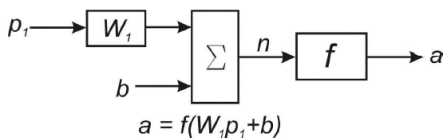


Рис. 2.1. Структура нейрона с единственным скалярным входом

Скалярный входной сигнал p умножается на скалярный весовой коэффициент W , и результирующий взвешенный вход $W \cdot p$ является аргументом функции активации нейрона f , которая порождает скалярный выход a .

Нейрон, показанный на рисунке 2.1, дополнен скалярным смещением b . Смещение суммируется со взвешенным входом $W \cdot p$ и приводит к сдвигу аргумента функции f на величину b . Действие смещения можно свести к схеме взвешивания, если представить, что нейрон имеет второй входной сигнал со значением, равным 1 ($b \cdot 1$). Вход n функции активации нейрона по-прежнему остается скалярным и равным сумме взвешенного входа и смещения b . Эта сумма ($W \cdot p + b \cdot 1$) является аргументом функции активации f , а выходом функции активации является сигнал a .

Константы W и b являются скалярными параметрами нейрона. Основной принцип работы нейронной сети состоит в настройке параметров нейрона таким образом, чтобы поведение сети соответствовало некоторому желаемому поведению. Регулируя веса и параметры смещения, можно обучить сеть выполнять конкретную работу; возможно также, что сеть сама будет корректировать свои параметры, чтобы достичь требуемого результата.

Уравнение нейрона со смещением имеет вид:

$$a = f(W \cdot p + b \cdot 1). \quad (2.1)$$

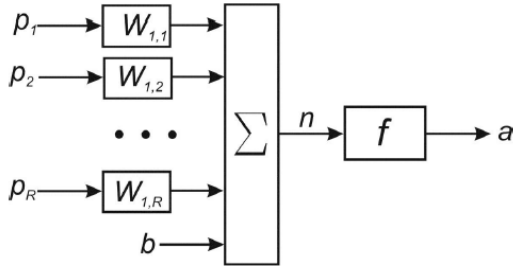
Как уже отмечалось, смещение b — настраиваемый скалярный параметр нейрона, который не является входом. В этом случае b — вес, а константа 1, которая управляет смещением, рассматривается как вход и может быть учтена в виде линейной комбинации векторов входа:

$$n = [W \ b] \begin{bmatrix} p \\ 1 \end{bmatrix} = W \cdot p + b \cdot 1. \quad (2.2)$$

2.2.2. Нейрон с векторным входом

Нейрон с одним вектором входа p с R элементами p_1, p_2, \dots, p_R показан на рисунке 2.2. Здесь каждый элемент входа умножается на ве-

са $W_{11}, W_{12}, \dots, W_{1R}$ соответственно, и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора строки W на вектор-столбец входа p .



$$a = f(\sum W_{i,j} p_j + b)$$

Рис. 2.2. Нейрон с одним вектором входа

Нейрон имеет смещение b , которое суммируется со взвешенной суммой входов. Результирующая сумма:

$$n = W_{11} \cdot p_1 + W_{12} \cdot p_2 + \dots + W_{1R} \cdot p_R + b \cdot 1 \quad (2.3)$$

или

$$n = W_{11} p_1 + W_{12} p_2 + \dots + W_{1R} p_R + b \quad (2.4)$$

и служит аргументом функции активации f .

В символах языка MATLAB это выражение записывается так:

$$n = W \cdot p + b. \quad (2.5)$$

Входом n функции активации нейрона служит сумма смещения b и произведение $W \cdot p$. Эта сумма преобразуется функцией активации f , на выходе которой получаем выход нейрона a , который в данном случае является скалярной величиной.

2.2.3. Функции активации

Функции активации (передаточные функции) нейрона могут иметь самый различный вид. Функция активации f , как правило, принадлежит к классу сигмоидальных функций, которые имеют две горизонтальные асимптоты и одну точку перегиба, с аргументом функции n

(входом) и значением функции (выходом) a . Рассмотрим три наиболее распространенные формы функции активации.

Единичная функция активации с жестким ограничением *hardlim*.

Эта функция описывается соотношением $a = \text{hardlim}(n) = 1(n)$ и показана на рисунке 2.3. Она равна 0, если $n < 0$, и равна 1, если $n \geq 0$. Чтобы построить график этой функции в диапазоне значений входа от -5 до $+5$, необходимо ввести следующие операторы языка MATLAB в командном окне:

```
>> n = -5:0.1:5;
plot(n,hardlim(n) , 'b+:') ;
```

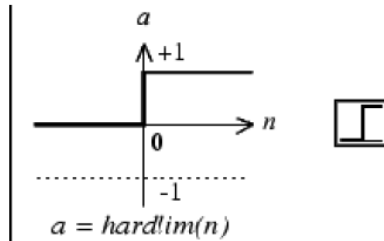


Рис. 2.3. Единичная функция активации с жестким ограничением *hardlim*

Линейная функция активации *purelin*.

Эта функция описывается соотношением $a = \text{purelin}(n) = n$ и показана на рисунке 2.4.

Чтобы построить график этой функции в диапазоне значений входа от -5 до $+5$, необходимо ввести следующие операторы языка MATLAB в командном окне:

```
>> n=-5:0.1:5;
plot(n,purelin(n) , 'b+:') ;
```

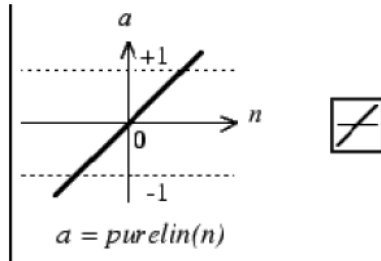



Рис. 2.4. Линейная функция активации purelin

Логистическая функция активации *logsig*.

Эта функция описывается соотношением $a = \text{logsig}(n) = 1/(1 + \exp(-n))$ и показана на рисунке 2.5. Данная функция принадлежит к классу сигмоидальных функций, и за ее аргумент может принимать любое значение в диапазоне от $-\infty$ до $+\infty$, а выход изменяется в диапазоне от 0 до 1. Благодаря свойству дифференцируемости (нет точек разрыва) эта функция часто используется в сетях с обучением на основе метода обратного распространения ошибки.

Чтобы построить график этой функции в диапазоне значений входа от -5 до $+5$, необходимо ввести следующие операторы языка MATLAB в командном окне:

```
>> n=-5:0.1:5;
plot(n,logsig(n), 'b+:' );
```

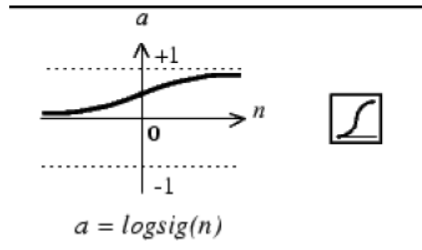


Рис. 2.5. Логистическая функция активации logsig

На укрупненной структурной схеме для обозначения типа функции активации применяются специальные графические символы, некоторые из них приведены на рисунке 2.6.

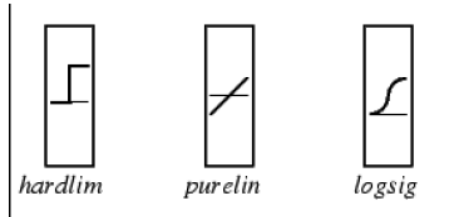


Рис. 2.6. Обозначения типа функции активации

Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации *hardlim* с жесткими ограничениями (рис. 2.7, использовано упрощенное изображение нейрона).

Каждое значение элемента вектора входа персептрона умножено на соответствующий вес W_{lj} , и сумма полученных взвешенных элементов является входом функции активации. Если вход функции активации $n \geq 0$, то нейрон персептрона возвращает 1, если $n < 0$, то 0.

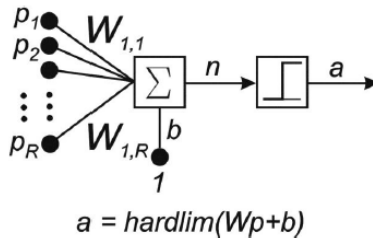


Рис. 2.7. Нейрон, используемый в модели персептрона

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на две области, как это показано на рисунке 2.8, для персептрона с двумя входами и смещением.

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

$$W_m \cdot p + b = 0. \quad (2.6)$$

Эта линия перпендикулярна к вектору весов W и смещена на величину b . Векторы входа выше линии L соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1; векторы входа ниже линии L соответствуют выходу персептрона, равному 0. При изменении значений смещения и весов граница линии L изменяет свое положение.

Персептрон без смещения всегда формирует разделяющую линию, проходящую через начало координат; добавление смещения формирует линию, которая не проходит через начало координат. В случае, когда размерность вектора входа превышает 2, т. е. входной вектор p имеет более 2 элементов, разделяющей границей будет служить гиперплоскость.

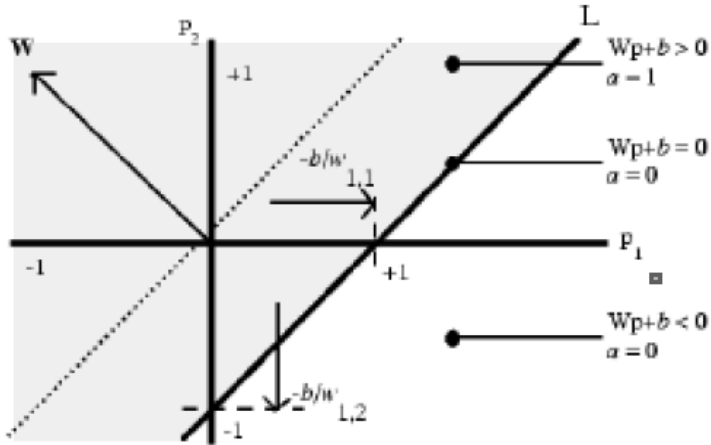


Рис. 2.8. Функция активации персептрона с двумя входами и смещением

2.2.4. Модель нейрона

Для выполнения работы запустите MATLAB и перейдите в Demos-режим:

Help → Demos → Toolboxes → Neural Networks →

Пример 1. Изучить простой нейрон.

Для ознакомления с *простым нейроном* выберите и запустите пример:

Simple neuron and transfer functions → Run Simple neuron and...

После запуска примера откроется окно, показанное на рисунке 2.9. В этом окне представлена схема нейрона, виртуальные органы для его настройки и графическая характеристика нейрона. Изменяя настройки нейрона и вид функции активации, можно изучить их влияние на свойства простого нейрона. Изменение настроек осуществляется перемещением соответствующих движков мышью.

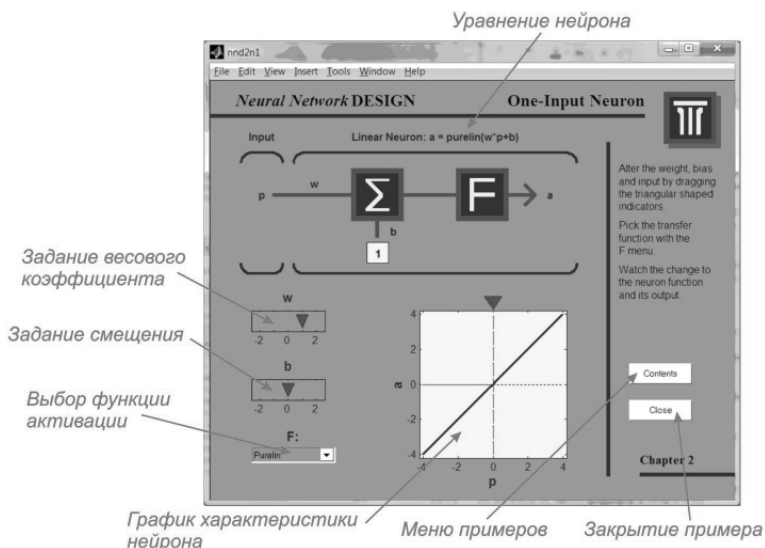


Рис. 2.9. Схема нейрона, виртуальные органы для его настройки и графическая характеристика нейрона

Вносимые изменения непосредственно отображаются на графике функции преобразования. При выборе разных функций активации вид графика и свойства нейрона существенно меняются.

Для завершения работы с примером щелкните кнопку *Close*.

Выйдите из *Demos*-режима и перейдите в командное окно MATLAB. Если в текущий момент это окно не отображается на экране, то настройте экран MATLAB, используя пункт меню *Desktop*. Дальнейшая работа выполняется в командном окне путем последовательного ввода команд и просмотра результатов.

§ 2.3. Персептрон

Рассмотрим персептрон на основе теоретических положений [2, 4, 6, 12–15].

Персептроном называется простейшая нейронная сеть, веса и смещения которой могут быть настроены таким образом, чтобы решить задачу классификации входных векторов. Задачи классификации позволяют решать сложные проблемы анализа коммутационных соединений, распознавания образов и других задач классификации с высоким быстродействием и гарантией правильного результата.

2.3.1. Архитектура сети

Персептрон состоит из единственного слоя, включающего S нейронов, как это показано на рисунке 2.10; веса W_{ij} — это коэффициенты передачи от j -го входа к i -му нейрону.

Структурная схема, приведённая на рисунке 2.10, называется слоем сети. Слой характеризуется матрицей весов W , смещением b , операциями умножения $W \cdot p$, суммирования и функцией активации f . Вектор входов p обычно не включается в характеристики слоя. Каждый раз, когда используется сокращенное обозначение сети, размерность матриц указывается под именами векторно-матричных переменных (см. рис. 2.9). Эта система обозначений поясняет строение сети и связанную с ней матричную математику.

Уравнение однослойного персептрона имеет вид:

$$\mathbf{a} = f(\mathbf{W} \cdot \mathbf{p} + \mathbf{b}), \quad (2.7)$$

где $\mathbf{a} = \text{hardlim}(\mathbf{W} \cdot \mathbf{p} + \mathbf{b})$.

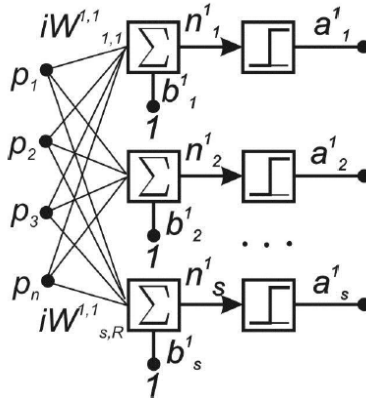


Рис. 2.10. Структурная схема персептрона

2.3.2. Моделирование персептрона средствами MATLAB

Для формирования модели однослойного персептрона в системе MATLAB предназначена функция **newp**.

Синтаксис:

`net = newp (PR,s,tf,lf)`

Описание:

Персептроны предназначены для решения задач классификации входных векторов, относящихся к классу линейно отделимых.

Функция `net = newp (PR, s, tf, lf)` формирует нейронную сеть персептрона.

Входные аргументы:

PR — массив размера $R \times 2$ минимальных и максимальных значений для R векторов входа;

s — число нейронов;

tf — функция активации из списка, по умолчанию `hardlim`;

lf — обучающая функция из списка, по умолчанию `learnp`.

Выходные аргументы:

net — объект класса network object.

Свойства:

Персептрон — это однослойная нейронная сеть с функциями взвешивания dotprod, накопления потенциала netsum и выбранной функцией активации. Слой характеризуется матрицей весов и вектором смещений, которые инициализируются М-функцией initzero.

Адаптация и обучение выполняются М-функциями adaptwb и trainwb, которые модифицируют значения весов и смещений до тех пор, пока не будет достигнуто требуемое значение критерия качества обучения в виде средней абсолютной ошибки, вычисляемой М-функцией mae.

Например, функция:

```
>> net = newp([0 2], 1);
```

создает персептрон с одноэлементным входом и одним нейроном; диапазон значений входа — [0 2]. В качестве функции активации персептрона по умолчанию используется функция hardlim.

Пример 2. Создать и изучить однослойный персептрон с одним двухэлементным вектором входа, значения элементов которого изменяются в диапазоне от -2 до 2 ($p_1 = [-2 \ 2]$, $p_2 = [-2 \ 2]$, число нейронов в сети $S = 1$).

Решение. Для создания персептрона в командном окне MATLAB задать:

```
>> clear, net = newp([-2 2;-2 2],1);  
%Создание персептрона net
```

По умолчанию веса и смещение равны нулю, и, для того чтобы установить желаемые значения, необходимо применить следующие операторы:

```
>> net.IW{1,1} = [-1 1]; % Веса w11= -1; w12  
= 1  
net.b{1} = [1]; % Смещение b = 1
```

Запишем уравнение (2.3) в развернутом виде для данной сети:

$$\begin{bmatrix} W_{11} \\ W_{12} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \end{bmatrix} + b_1 = 0$$

$$\begin{bmatrix} -1 \\ 1 \end{bmatrix} \begin{bmatrix} p_1 & p_2 \end{bmatrix} + 1 = 0$$

В этом случае разделяющая линия имеет вид

$$L: -p_1 + p_2 + 1 = 0$$

и соответствует линии L на рисунке 2.8.

Определим реакцию сети на входные векторы p_1 и p_2 , расположенные по разные стороны от разделяющей линии. Для чего последовательно введем $p_1 = [1; 1]$ и $p_2 = [1; -1]$ как указано ниже:

```
>> p1 = [1; 1];
a1 = sim(net,p1)
% Моделирование сети net с входным вектором
p1
```

```
a1 =
1
```

```
>> p2 = [1; -1];
a2 = sim(net,p2)
% Моделирование сети net с входным вектором
p2
```

```
a2 =
0
```

В результате получили $a_1 = 1$ и $a_2 = 0$, т. е. персептрон правильно классифицировал эти два вектора.

Заметим, что можно было бы ввести последовательность двух векторов в виде массива ячеек p_3 и получить результат также в виде массива ячеек a_3 :

```
>> p3 = { [1; 1] [1; -1] }
```

```
p3 =
```



```
[2x1 double] [2x1 double]
```

```
>> a3 = sim(net,p3)
```

```
% Моделирование сети net при входном сигнале  
p3
```

```
a3 =
```

```
[1] [0]
```

Пример 3. Создать и изучить однослойный персептрон с одним трехэлементным вектором входа, значения элементов которого изменяются в диапазоне от -2 до 2 ($p_1 = [-2 \ 2]$, $p_2 = [-2 \ 2]$, $p_3 = [-2 \ 2]$, число нейронов в сети $S = 2$).

Решение. Для создания персептрона в командном окне MATLAB задать:

```
>> clear, net = newp([-2 2;-2 2;-2 2],2);
```

```
%Создание персептрона net
```

По умолчанию веса и смещение равны нулю, и для того, чтобы установить желаемые значения, необходимо применить следующие операторы:

```
>> net.IW{1,1} = [3 1 2; 1 4 0];
```

```
% Веса w11= 3; w12 = 1; w13 = 2
```

```
Веса w21= 1; w22 = 4; w23 = 0
```

```
>> net.b{1} = [2;5];
```

```
% Смещение b1 = 2; b2 = 5;
```

В этом случае разделяющая линия для каждого нейрона имеет вид:

$$L: W_{11} \cdot p_1 + W_{12} \cdot p_2 + W_{13} \cdot p_3 + b = 0$$

$$L \text{ (1 нейрон): } 3 \cdot p_1 + 1 \cdot p_2 + 2 \cdot p_3 + 2 = 0$$

$$L \text{ (2 нейрон): } 1 \cdot p_1 + 4 \cdot p_2 + 0 \cdot p_3 + 5 = 0$$

Определим реакцию сети на входные векторы, расположенные по разные стороны от разделяющей линии.

```
>> p1 = [1; 3; 2];
a1 = sim(net,p1)

a1 =
     1
     1

>> p1 = [1; 3; -5];
a1 = sim(net,p1)

a1 =
     0
     1
```

Пример 4. Изучить нейрон с векторным входом.

Для выполнения работы запустите MATLAB и перейдите в Demos-режим:

Help → Demos → Toolboxes → Neural Networks →

Для изучения свойств нейрона с векторным входом необходимо запустить пример:

Neuron with vector input → Run Neuron with vector...

Окно примера подобно окну простого нейрона из примера 1 и отличается большим количеством настроек. Пример позволяет изучить однослойный персептрон с одним нейроном в сети и с одним двух-элементным вектором входа.

Необходимо изучить свойства нейрона с векторным входом и влияние разных настроек на характеристику нейрона из примера 2.

Рассмотрим персептрон с весами $w_{11} = -1$; $w_{12} = 1$ и смещением $b = 1$. Если входной вектор $p_1 = [1; 1]$, то реакция персептрона $a = f(W \cdot p + b) > 0$ (т. е. нейрон персептрона возвращает 1) (рис. 2.11), если входной вектор $p_1 = [1; -1]$, то реакция персептрона $a = f(W \cdot p + b) < 0$ (т. е. нейрон персептрона возвращает 0) (рис. 2.12).

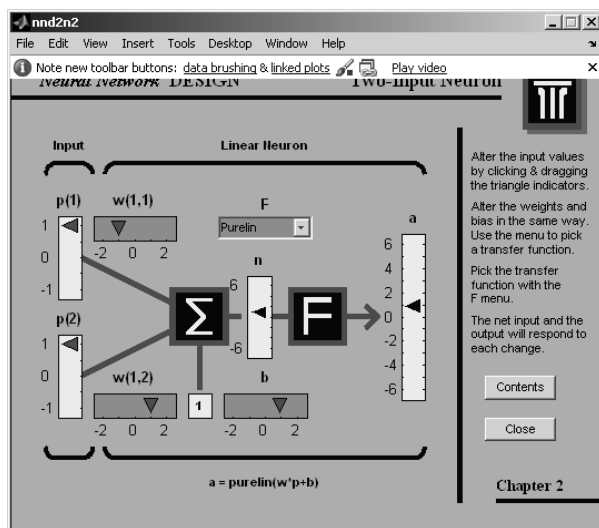


Рис. 2.11. Схема нейрона с векторным входом, веса $w_{11} = -1$; $w_{12} = 1$, смещение $b = 1$, входной вектор $p_1 = [1; 1]$

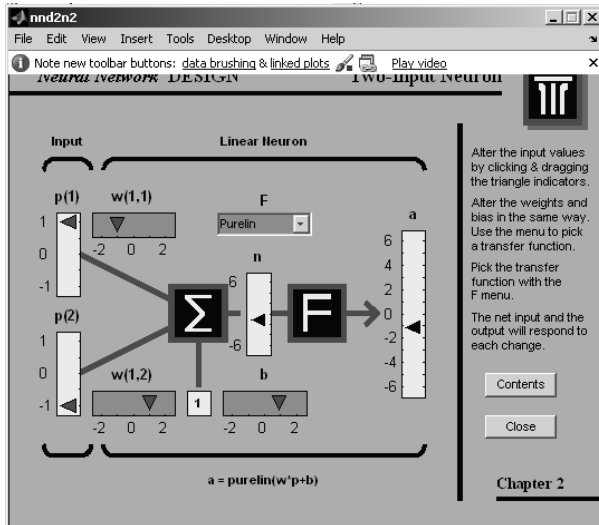


Рис. 2.12. Схема нейрона с векторным входом, веса $w_{11} = -1$; $w_{12} = 1$, смещение $b = 1$, входной вектор $p_1 = [1; -1]$

2.3.3. Инициализация параметров

Для однослойного персептрона в качестве параметров нейронной сети в общем случае выступают веса входов и смещения.

Пример 5. Допустим, что создается персептрон с двухэлементным вектором входа и одним нейроном

```
>> clear, net = newp([-2 2;-2 2],1);
```

Необходимо определить характеристики весов входа и переустановить значения элементов матрицы весов и смещения.

Решение. Запросим характеристики весов входа и получим ответ *ans* =:

```
>> net.inputweights{1, 1}
```

```
ans =
delays: 0
initFcn: 'initzero'
learn: 1
learnFcn: 'learnp'
learnParam: []
size: [1 2]
userdata: [1x1 struct]
weightFcn: 'dotprod'
```

Из этого списка следует, что в качестве функции инициализации по умолчанию используется функция *initzero*, которая присваивает весам входа нулевые значения. В этом можно убедиться, если извлечь значения элементов матрицы весов и смещения:

```
>> wts = net.IW{1,1},
bias = net.b{1}
```

```
wts =
0 0
bias =
0
```

Теперь переустановим значения элементов матрицы весов и смещения:

```
>> net.IW{1,1} = [3, 4]; net.b{1} = 5;
wts = net.IW{1,1}, bias = net.b{1}

wts =
3     4
bias =
5
```

Для того чтобы вернуться к первоначальным установкам параметров персептрона, предназначена функция *init*:

```
>> net = init(net); wts = net.IW{1,1}, bias =
net.b{1}

wts =
0     0
bias =
0
```

Пример 6. Изменить способ инициализации персептрона.

Решение. Можно изменить способ, каким инициализируется персептрон с помощью функции *init*. Для этого достаточно изменить тип функций инициализации, которые применяются для установки первоначальных значений весов входов и смещений.

Например, воспользуемся функцией инициализации *rands*, которая устанавливает случайные значения параметров персептрона:

```
>> net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
% Задать функции инициализации
весов и смещений

net = init(net);
```

```
% Выполнить инициализацию ранее созданной се-
ти с новыми функциями
wts = net.IW{1,1}, bias = net.b{1}
```

```
wts =
-0.1886  0.8709
bias =
-0.6475
```

Видно, что веса и смещения выбраны случайным образом.

§ 2.4. Задание для лабораторной работы

Задание 1. Изучить простой нейрон, для чего выполнить действия, указанные в п. 2.2.4 в **примере 1**. Изменяя настройки нейрона и вид функции активации, изучить их влияние на свойства простого нейрона. Результаты внести в отчет в виде картинок изображений, полученных в MATLAB.

Задание 2. Изучить нейрон с векторным входом, для чего выполнить действия, указанные в п. 2.3.2 в **примере 4**. Изменяя настройки нейрона и вид функции активации, изучить их влияние на свойства нейрона с векторным входом. Результаты внести в отчет в виде картинок изображений, полученных в MATLAB.

Задание 3. Создать и изучить однослойный персептрон, для чего выполнить действия, указанные п. 2.3.2 **примеров 2, 3**. Результаты внести в отчет в виде картинок изображений, полученных в командном окне MATLAB.

Задание 4. Провести инициализацию параметров персептрона с двухэлементным вектором входа и одним нейроном по п. 2.3.3 (**примеры 5, 6**). Результаты внести в отчет в виде картинок изображений, полученных в командном окне MATLAB.

Задание 5.

1. Создать и изучить однослойный персептрон по исходным данным из таблицы 2.1.

2. Определить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5).

3. Переустановить значения матриц весов и смещений с помощью рассмотренных функций инициализации. Вариант задания указывает преподаватель.

Таблица 2.1

Варианты задания

№ варианта	Число входов	Пределы изменения входов	Нейронов в слое
1	2	-1...1	2
2	2	-3...3	2
3	2	-9...9	3
4	2	-5...5	3
5	2	-8...8	2
6	2	-1...1	3
7	2	-2...2	3
8	2	-4...4	2
9	2	-6...6	3
10	2	-4...4	3

Указание. Все решения по заданиям производить с использованием системы MATLAB. Для оформления работы использовать Microsoft WORD.

§ 2.5. Структура отчета

По материалам работы каждым студентом составляется отчет по установленной форме с использованием Microsoft WORD. Расчеты, диаграммы, графики следует выполнять с использованием ПЭВМ. Особое внимание при оформлении отчета студенты должны обратить на составление выводов по выполненной работе. В выводах нужно сопоставить результаты проведенных исследований с известными из теоретического курса закономерностями и выяснить согласованность полученных результатов с теоретическими. Полностью оформленный

исполнителем отчет представляется каждым студентом преподавателю на следующем занятии.

Отчет по лабораторной работе должен содержать:

- 1) титульный лист, оформленный по образцу;
- 2) цель лабораторной работы и исходные данные в Microsoft WORD;
- 3) результаты исследований по заданиям 1–3 в виде таблиц, рисунков, графиков в Microsoft WORD;
- 4) результаты исследований по заданию 4 должны кроме того содержать: структурную схему нейронной сети; алгоритм, текст программы и график;
- 5) выводы по заданиям.

К отчету прилагаются программы в системе MATLAB в электронном виде.

Представленные в отчете результаты, порядок их получения, расчетов, графиков и диаграмм студенты обязаны уметь четко пояснить. За проведенную работу и оформленный отчет преподаватель выставит дифференцированную оценку.

Глава 3

ЛАБОРАТОРНАЯ РАБОТА

«ЛИНЕЙНЫЕ НЕЙРОННЫЕ СЕТИ.

ОБУЧЕНИЕ ЛИНЕЙНОЙ СЕТИ В MATLAB»

§ 3.1. Цель работы

Лабораторная работа выполняется на основе теоретических положений искусственного интеллекта на ПК с использованием систем MatLab (MATLAB) и Microsoft WORD в среде Windows XP/8/10.

Цели работы:

- 1) изучение модели нейрона и архитектуры линейной нейронной сети;
- 2) изучение процедуры настройки параметров линейных нейронных сетей посредством прямого расчета в системе MATLAB;
- 3) изучение алгоритма настройки параметров линейных нейронных сетей с помощью процедуры обучения train в системе MATLAB;
- 4) получение умений и навыков:
 - создания и исследования моделей линейных нейронных сетей в системе MATLAB;
 - решения задач классификации с помощью линейной нейронной сети;
 - анализа полученных результатов.

§ 3.2. Линейные сети

Рассмотрим модели нейрона и архитектуры линейной нейронной сети, процедуры настройки параметров линейных нейронных сетей на основе положений работ [2, 6, 12–19].

3.2.1. Структура линейной сети

Линейные сети по своей структуре аналогичны персептрону и отличаются лишь функцией активации. Выход линейной сети может

принимать любое значение, в то время как выход персептрона ограничен значениями 0 или 1.

На рисунке 3.1 показан линейный нейрон с двумя входами. Он имеет структуру, сходную со структурой персептрона. Единственное отличие состоит в том, что используется линейная функция активации *purelin*.

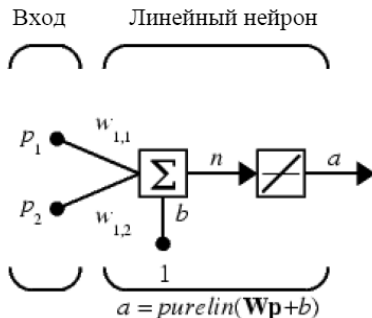


Рис. 3.1. Линейный нейрон с двумя входами

Весовая матрица \mathbf{W} в этом случае имеет только одну строку, и выход сети определяется выражением

$$\mathbf{a} = \text{purelin}(\mathbf{n}) = \text{purelin}(\mathbf{W}\mathbf{p} + \mathbf{b}) = \mathbf{W}\mathbf{p} + \mathbf{b} = w_{11}p_1 + w_{12}p_2 + b. \quad (3.1)$$

Подобно персептрону, линейная сеть задает в пространстве входов разделяющую линию, на которой функция активации n равна 0 (рис. 3.2).

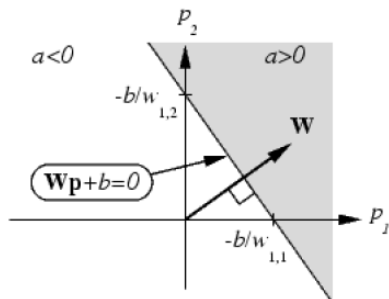


Рис. 3.2. Линейная сеть в пространстве входов

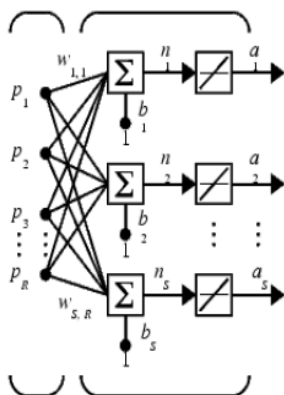
Векторы входа, расположенные выше этой линии, соответствуют положительным значениям выхода, а расположенные ниже — отрицательным. Это означает, что линейная сеть может быть применена для решения задач классификации.

Однако такая классификация может быть выполнена только для класса линейно отделимых объектов. Таким образом, линейные сети имеют то же самое ограничение, что и персептрон.

3.2.2. Архитектура сети

Линейная сеть, показанная на рисунке 3.3а, включает S нейронов, размещенных в одном слое и связанных с R входами через матрицу весов \mathbf{W} . На рисунке 3.3б показана укрупненная структурная схема этой сети, вектор выхода в которой имеет размер $S \times 1$.

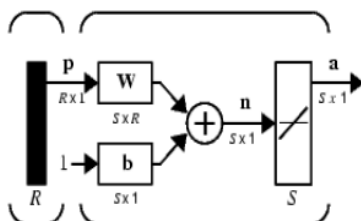
Входы Слой линейных нейронов



$$\mathbf{a} = \text{purelin}(\mathbf{IW}^{11}\mathbf{p} + \mathbf{b}^1)$$

а

Входы Слой линейных нейронов



$$\mathbf{a} = \text{purelin}(\mathbf{IW}^{11}\mathbf{p} + \mathbf{b}^1)$$

б

Рис. 3.3. Линейная сеть:

а) с S -нейронами, размещенными в одном слое;

б) укрупненная структурная схема этой сети.

3.2.3. Создание линейной сети

Для создания модели линейной сети используют функцию **newlin**.

Синтаксис:

```
net = newlin(PR,s,id,lr)
```

```
net = newlin(PR,s,0,P)
```

Описание:

Линейные слои находят применение при решении задач аппроксимации, фильтрации и предсказания сигналов, построении моделей динамических систем в задачах управления.

Функция $net = newlin(PR, s, id, lr)$ формирует нейронную сеть в виде линейного слоя.

Входные аргументы:

PR — массив размера $R \times 2$ минимальных и максимальных значений для R векторов входа;

s — число нейронов;

id — описание линии задержки на входе сети, по умолчанию [0];

lr — параметр скорости настройки, по умолчанию 0.01.

Выходные аргументы:

net — объект класса network object с архитектурой линейного слоя.

Функция $net = newlin(PR, s, 0, P)$, где P — матрица векторов входа, формирует линейный слой с параметром скорости настройки, гарантирующим максимальную степень устойчивости слоя для данного входа P .

Например, линейный слой, который для заданного входа воспроизводит заданный отклик системы, имеет вид:

```
net = newlin([-1 1], 1, [0 1 2], 0.01)
```

Тогда архитектура линейного слоя: линия задержки типа [0 1 2], 1 нейрон, вектор входа с элементами из диапазона [-1 1], параметр скорости настройки – 0.01.

Алгоритм:

Линейный слой использует функцию взвешивания dotprod, функцию накопления потенциала netsum и функцию активации purelin.

Слой характеризуется матрицей весов и вектором смещений, которые инициализируются М-функцией `initzero`.

Адаптация и обучение выполняются М-функциями `adaptwb` и `trainwb`, которые модифицируют веса и смещения, используя М-функцию `learnwb`, до тех пор, пока не будет достигнуто требуемое значение критерия качества обучения в виде средней квадратичной ошибки, вычисляемой М-функцией `mse`.

Пример 1. Линейную сеть с одним нейроном, показанную на рис. 3.1, можно создать следующим образом:

```
>> clear, net = newlin([-1 1; -1 1],1);
```

Первый входной аргумент задает диапазон изменения элементов вектора входа; второй аргумент указывает, что сеть имеет единственный выход. Начальные веса и смещение по умолчанию равны нулю. Присвоим весам и смещению следующие значения:

```
>> net.IW{1,1} = [2 3];  
net.b{1} = [-4]
```

Теперь можно промоделировать линейную сеть для следующего предъявленного вектора входа:

```
>> p = [5;6];  
a = sim(net,p)
```

```
a =
```

24

Видно, что сеть правильно классифицировала входной вектор.

3.2.4. Конструирование линейной сети

Для конструирования модели линейной сети используют функцию `newlind`.

Синтаксис:

```
net = newlind(P, T)
```

Описание:

Линейный слой LIND использует для расчета весов и смещений процедуру решения систем линейных алгебраических уравнений на основе метода наименьших квадратов, и поэтому в наибольшей степени он приспособлен для решения задач аппроксимации, когда требуется подобрать коэффициенты аппроксимирующей функции. В задачах управления такой линейный слой можно применять для идентификации параметров динамических систем.

Функция `net = newlind(P, T)` формирует нейронную сеть, используя только обучающие последовательности входа P размера $R \times Q$ и цели T размера $S \times Q$. Выходом является объект класса `network object` с архитектурой линейного слоя.

Алгоритм:

Функция `newlind` вычисляет значения веса W и смещения b для линейного уровня с входом P и целью T , решая линейное уравнение в смысле метода наименьших квадратов:

$$[W \ b] * [P; \text{ones}] = T.$$

Пример 2. Требуется сформировать линейный слой, который обеспечивает для заданного входа P выход, близкий к цели T , если заданы следующие обучающие последовательности:

$$P = 0:3; \quad T = [0.0 \ 2.0 \ 4.1 \ 5.9].$$

Анализ данных подсказывает, что требуется найти аппроксимирующую кривую, которая близка к зависимости $t = 2p$. Применение линейного слоя LIND в данном случае вполне оправдано.

```
>> P = 0:3;
T = [0.0 2.0 4.1 5.9];
net = newlind(P, T);
net.IW, net.b
```

```
ans =      1.9800
ans =      0.3000
```

Соответствующая аппроксимирующая кривая описывается соотношением

$$y_k = 1.9800r_k + 0.3000. \quad (3.2)$$

Выполним моделирование сформированного линейного слоя:

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

```
Y =    0.0300    2.0100    3.9900    5.9700
```

```
>>Y = sim (net,0:0.5:3)
```

```
Y =    0.0300    1.0200    2.0100    3.0000
3.9900    4.9800    5.9700
```

Рассмотрим задачу восстановления некоторой, вообще говоря, неизвестной зависимости по имеющимся экспериментальным данным с использованием линейной НС.

Пример 3. Пусть экспериментальная информация задана значениями $x = [1.0 \ 1.5 \ 3.0 \ -1.2]$, $y = [0.5 \ 1.1 \ 3.0 \ -1.0]$. Необходимо восстановить зависимость по имеющимся экспериментальным данным с использованием НС.

Решение. Создадим векторы входа и целей:

```
>> x = [1.0 1.5 3.0 -1.2];
```

```
>> y = [0.5 1.1 3.0 -1.0];
```

Теперь создадим линейную нейронную сеть:

```
>> b=newlind(x,y); % Создание линейной НС с именем b
```

Проведем опрос сети для значения входа, равного 3.0 (этому, согласно экспериментальным данным, соответствует целевое значение 3.0):

```
>> y1 = sim(b, 3.0) % Опрос сети
```

```
y1 =
    2.7003
```

Погрешность восстановления по данным обучающей выборки в данном случае — 10%.

§ 3.3. Процедуры настройки линейной сети

3.3.1. Процедура настройки линейной сети с использованием функции adapt

Множественно используя функции `sim` и `learnpr` для изменения весов и смещения персептрона, можно в конечном счете построить разделяющую линию, которая решит задачу классификации при условии, что персептрон может решать ее. Каждая реализация процесса настройки с использованием всего обучающего множества называется проходом, или циклом. Такой цикл может быть выполнен с помощью специальной функции адаптации `adapt`. При каждом проходе функция `adapt` использует обучающее множество, вычисляет выход, погрешность и выполняет подстройку параметров персептрона.

Процедура адаптации не гарантирует, что синтезированная сеть выполнит классификацию нового вектора входа. Возможно, потребуется новая настройка матрицы весов W и вектора смещений b с использованием функции `adapt`.

Чтобы пояснить процедуру адаптации, рассмотрим простой пример. Выберем персептрон с одним нейроном и двухэлементным вектором входа.

Эта сеть достаточно проста, так что все расчеты можно выполнить вручную.

Предположим, что можно с помощью персептрона решить задачу классификации векторов, если задано следующее обучающее множество:

$$\{p_1 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, t_1 = 0\} \{p_2 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, t_2 = 1\} \{p_3 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, t_3 = 0\} \{p_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, t_4 = 1\}$$

Используем нулевые начальные веса и смещения. Для обозначения переменных каждого шага используем круглые скобки. Таким образом, начальные значения вектора весов $wT(0)$ и смещения $b(0)$ соответственно равны $wT(0) = [0 \ 0]$ и $b(0) = 0$.

1-й шаг процедуры адаптации

Вычислим выход персептрона для первого вектора входа p_1 , используя начальные веса и смещение:

$$a = \text{hardlim}(\mathbf{w}^T(0)\mathbf{p}_1 + b(0)) = \text{hardlim}\left(\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + 0\right) = \text{hardlim}(0) = 1.$$

Выход не совпадает с целевым значением t_1 , поэтому необходимо применить правило настройки (обучения) персептрона, чтобы вычислить требуемые изменения весов и смещений:

$$\begin{cases} e = t_1 - a = 0 - 1 = -1 \\ \Delta \mathbf{w}^T = e \mathbf{p}^T = (-1)[2 \ 2] = [-2 \ -2] \\ \Delta b = e = -1 \end{cases}$$

Вычислим новые веса и смещение, используя введенные ранее правила обучения персептрона.

$$\begin{cases} \mathbf{w}^{T\text{new}} = \mathbf{w}^{T\text{old}} + \Delta \mathbf{w}^T = [0 \ 0] + [-2 \ -2] = \mathbf{w}^T(1) \\ b^{\text{new}} = b^{\text{old}} + \Delta b = 0 + (-1) = b(1) \end{cases}$$

2-й шаг процедуры адаптации

Обратимся к новому вектору входа p_2 , тогда:

$$a = \text{hardlim}(\mathbf{w}^T(1)\mathbf{p}_2 + b(1)) = \text{hardlim}\left(\begin{bmatrix} -2 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} + (-1)\right) = \text{hardlim}(1) = 1.$$

В этом случае выход персептрона совпадает с целевым выходом, так что погрешность равна 0 и не требуется изменений в весах или смещении. Таким образом:

$$\begin{cases} \mathbf{w}^T(2) = \mathbf{w}^T(1) = [-2 \ -2] \\ b(2) = b(1) = -1 \end{cases}$$

3-й шаг процедуры адаптации

Продолжим этот процесс и убедимся, что после третьего шага настройки не изменились:

$$\begin{cases} \mathbf{w}^T(3) = \mathbf{w}^T(2) = [-2 \ -2] \\ b(3) = b(2) = -1 \end{cases}$$

4-й шаг процедуры адаптации

После четвертого примем значение:

$$\begin{cases} \mathbf{w}^T(4) = [-3 \ -1] \\ b(4) = 0 \end{cases}$$

Чтобы определить, получено ли удовлетворительное решение, требуется сделать один проход через все векторы входа с целью проверить, соответствуют ли решения обучающему множеству.

5-й шаг процедуры адаптации

Вновь используем первый член обучающей последовательности и получаем:

$$\begin{cases} \mathbf{w}^T(5) = \mathbf{w}^T(4) = [-3 \ -1] \\ b(5) = b(4) = 0 \end{cases}$$

6-й шаг процедуры адаптации

Переходя ко второму члену, получим следующий результат:

$$\begin{cases} \mathbf{w}^T(6) = [-2 \ -3] \\ b(6) = 1 \end{cases}$$

Этим заканчиваются ручные вычисления.

Пример 4. Для решения этой задачи в MATLAB сформируем модель персептрона:

```
>> clear, net = newp([-2 2;-2 2],1);
```

Введем первый элемент обучающего множества:

```
>> p = { [2; 2] }; t = { 0 };
```

Установим параметр **passes** (число проходов) равным 1 и выполним один шаг настройки:

```
>> net.adaptParam.passes = 1;
[net,a,e] = adapt(net,p,t); a,e
```

```
a = [1]
```

```
e = [-1]
```

Скорректированные вектор весов и смещение определим следующим образом:

```
>> twts = net.IW{1,1}, tbiase = net.b{1}
```

```
twts =
```

```
-2    -2
```

```
tbiase =
```

```
-1
```

Это совпадает с результатами, полученными при ручном расчете. Теперь можно ввести второй элемент обучающего множества и т. д., то есть повторить всю процедуру ручного счета и получить те же результаты.

Но можно эту работу выполнить автоматически, задав сразу всё обучающее множество и выполнив один проход:

```
>> clear, net = newp([-2 2;-2 2],1);
net.trainParam.passes = 1;
p = {[2;2] [1;-2] [-2;2] [-1;1]};
t = {0 1 0 1};
```

Теперь обучим сеть.

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

Возвращаются выход и ошибка

```
>> a =
[1] [1] [0] [0]
e =
[-1] [0] [0] [1]
```

Скорректированные вектор весов и смещение определяем следующим образом:

```
>> twts = net.IW{1,1}, tbiase = net.b{1}
twts =
-3 -1
tbiase =
0
```

Моделируя полученную сеть по каждому входу, получим

```
>> a1 = sim(net,p)
a1 =
[0] [0] [1] [1]
```

Можно убедиться, что не все выходы равны целевым значениям обучающего множества. Это означает, что следует продолжить настройку персептрона.

Выполним еще один цикл настройки:

```
>> [net,a,e] = adapt(net,p,t); a, e
a =
[0] [0] [0] [1]
e =
[0] [1] [0] [0]

>> twts = net.IW{1,1}, tbiase = net.b{1}
twts =
-2 -3 tbiase =
1
a1 = sim(net,p)
a1 =
[0] [1] [0] [1]
```

Теперь решение совпадает с целевыми выходами обучающего множества, и все входы классифицированы правильно.

Если бы рассчитанные выходы персептрона не совпали с целевыми значениями, то необходимо было бы выполнить еще несколько циклов настройки, применяя функцию `adapt` и проверяя правильность получаемых результатов.

Итак, для настройки (обучения) персептрона применяется процедура адаптации, которая корректирует параметры персептрона по результатам обработки каждого входного вектора. Применение функции `adapt` гарантирует, что любая задача классификации с линейно отделимыми векторами будет решена за конечное число циклов настройки.

Нейронные сети на основе персептрона имеют ряд ограничений. Во-первых, выход персептрона может принимать только одно из двух значений (0 или 1); во-вторых, персептроны могут решать задачи классификации только для линейно отделимых наборов векторов. Ес-

ли векторы входа линейно неотделимы, то процедура адаптации не в состоянии классифицировать все векторы должным образом.

Для решения более сложных задач можно использовать сети с несколькими персептронами. Например, для классификации четырех векторов на четыре группы можно построить сеть с двумя персептронами, чтобы сформировать две разделяющие линии и таким образом приписать каждому вектору свою область.

3.3.2. Процедура настройки линейной сети с использованием функции *newlind*

Линейные сети, как и персептроны, способны решать только линейно отделимые задачи классификации, однако в них используется другое правило обучения, основанное на методе обучения наименьших квадратов, которое является более мощным, чем правило обучения персептрона. Для заданной линейной сети и соответствующего множества векторов входа и целей можно вычислить вектор выхода сети и сформировать разность между вектором выхода и целевым вектором, которая определит некоторую погрешность. В процессе обучения сети требуется найти такие значения весов и смещений, чтобы сумма квадратов соответствующих погрешностей была минимальной, поэтому настройка параметров выполняется таким образом, чтобы обеспечить минимум ошибки. Эта задача разрешима, так как для линейной сети поверхность ошибки как функция входов имеет единственный минимум и отыскание этого минимума не вызывает трудностей.

Как и для персептрона, для линейной сети применяется процедура обучения с учителем, которая использует обучающее множество вида:

$$\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_Q, t_Q\}. \quad (3.3)$$

Требуется минимизировать одну из следующих функций квадратичной ошибки:

$$mse = \frac{1}{Q} \sum_{k=1}^Q e(k)^2 = \frac{1}{Q} \sum_{k=1}^Q (t(k) - a(k))^2 \quad (3.4)$$

или

$$sse = \sum_{k=1}^Q e(k)^2 = \sum_{k=1}^Q (t(k) - a(k))^2, \quad (3.5)$$

где mse — средняя квадратичная ошибка; sse — сумма квадратов ошибок.

В отличие от многих других сетей настройка линейной сети для заданного обучающего множества может быть выполнена посредством прямого расчета с использованием *newlind*, т. е. можно построить поверхность ошибки и найти на этой поверхности точку минимума, которая будет соответствовать оптимальным весам и смещениям для данной сети. Проиллюстрируем это на следующем примере. Предположим, что заданы следующие векторы, принадлежащие обучающему множеству $P=[1 \ -1.2]$; $T=[0.5 \ 1]$.

Структурная схема этого линейного нейрона представлена на рисунке 3.1. Запишем уравнение выхода нейрона:

$$a = \text{purelin}(n) = \text{purelin}(wp+b) = wp+b \quad (3.6)$$

Графическая интерпретация настройки веса и смещения для данного нейрона при двух обучающих множествах сводится к построению прямой, проходящей через две заданные точки, и представлена на рисунке 3.4.

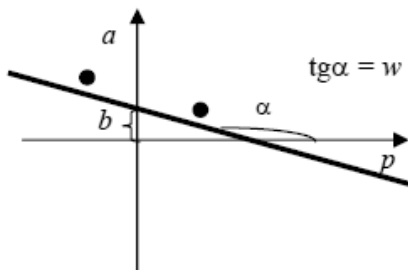


Рис. 3.4. Графическая интерпретация настройки веса и смещения для нейрона

Пример 5. Построим линейную сеть и промоделируем ее:

```
>> clear, P=[1 -1.2];T=[0.5 1]
net = newlind(P,T);

Y = sim(net, P)
Y =
           0.5000           1.0000
>> net.IW{1,1}

ans =
      -0.2273

net.b
ans =
      [0.7273]
```

Выход сети соответствует целевому вектору, т.е. оптимальными весом и смещением нейрона будут $w = -0,2273$; $b = 0,7273$.

Зададим следующий диапазон весов и смещений:

```
>> w_range=-1:0.1: 0;
b_range=0.5:0.1:1;
```

Рассчитаем критерий качества обучения:

```
>> ES = errsurf(P,T, w_range, b_range,
'purelin');
```

Построим линии уровня поверхности функции критерия качества обучения в пространстве параметров сети (рис. 3.5):

```
>> contour(w_range, b_range, ES, 20)
hold on

>> plot(-2.273e-001, 7.273e-001, 'x')
hold off
```

На графике знаком «х» отмечены оптимальные значения веса и смещения для данной сети.

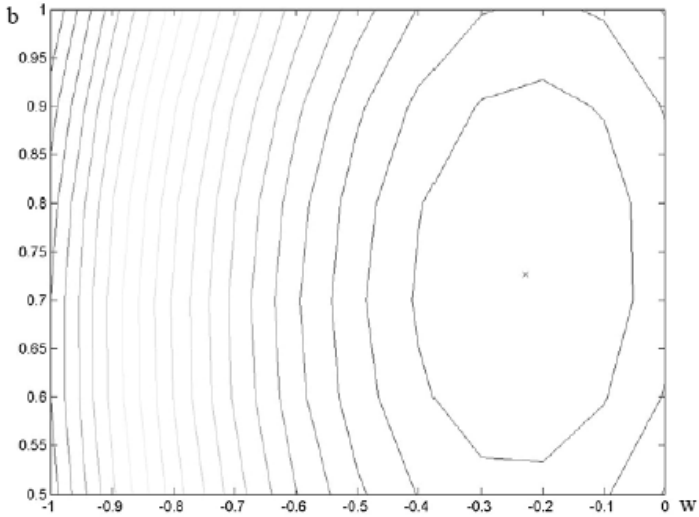


Рис. 3.5. Линии уровня поверхности функции критерия качества обучения в пространстве параметров сети

§ 3.4. Обучение линейной сети. Обучающее правило наименьших квадратов

Для линейной нейронной сети используется рекуррентное обучающее правило наименьших квадратов, которое является более мощным, чем обучающее правило персептрона. Правило наименьших квадратов, или правило обучения WH (Уидроу — Хоффа), минимизирует среднее значение суммы квадратов ошибок обучения. Процесс обучения нейронной сети состоит в следующем. Авторы алгоритма предположили, что можно оценивать полную среднюю квадратичную погрешность, используя среднюю квадратичную погрешность на каждой итерации.

Сформируем частную производную по весам и смещению от квадрата погрешности на k -й итерации:

$$\begin{cases} \frac{\partial e^2(k)}{\partial w_{1,j}} = 2e(k) \frac{\partial e(k)}{\partial w_{1,j}}, & j = 1, \dots, R; \\ \frac{\partial e^2(k)}{\partial b} = 2e(k) \frac{\partial e(k)}{\partial b}. \end{cases} \quad (3.7)$$

Подставляя выражение для ошибки в форме:

$$e(k) = t(k) - \left(\sum_{j=1}^R w_{1,j} p_j(k) + b \right), \quad (3.8)$$

Получим

$$\begin{cases} \frac{\partial e(k)}{\partial w_{1,j}} = -p_j(k); \\ \frac{\partial e(k)}{\partial b} = -1. \end{cases} \quad (3.9)$$

Здесь $p_j(k)$ — j -й элемент вектора входа на k -й итерации. Эти соотношения лежат в основе обучающего алгоритма WH:

$$\begin{cases} \mathbf{w}(k+1) = \mathbf{w}(k) + e(k) \mathbf{p}^T(k); \\ b(k+1) = b(k) + 2\alpha e(k). \end{cases} \quad (3.10)$$

Результат может быть обобщен на случай многих нейронов и представлен в следующей матричной форме:

$$\begin{cases} \mathbf{W}(k+1) = \mathbf{W}(k) + \mathbf{e}(k) \mathbf{p}^T(k); \\ \mathbf{b}(k+1) = \mathbf{b}(k) + 2\alpha \mathbf{e}(k). \end{cases} \quad (3.11)$$

Здесь ошибка \mathbf{e} и смещение \mathbf{b} — векторы; α — параметр скорости обучения.

При больших значениях α обучение происходит быстро, однако при очень больших значениях может приводить к неустойчивости. Чтобы гарантировать устойчивость процесса обучения, параметр скорости обучения не должен превышать величины $1/\max(|\lambda|)$, где λ — собственное значение матрицы корреляций $\mathbf{p}^* \mathbf{p}^T$ векторов входа.

Используя правило обучения WH и метод наискорейшего спуска, всегда можно обучить сеть так, чтобы погрешность обучения была минимальной.

Функция **learnwh** предназначена для настройки параметров линейной сети и реализует следующее обучающее правило:

$$\begin{cases} \mathbf{dw} = lr * \mathbf{e} * \mathbf{p}^T; \\ \mathbf{db} = lr * \sum_{i=1}^Q e_i, \end{cases} \quad (3.12)$$

где lr — параметр скорости обучения. Максимальное значение параметра скорости обучения, которое гарантирует устойчивость процедуры настройки, вычисляется с помощью функции **maxlinlr**.

С помощью демонстрационной программы **demolin7** можно исследовать устойчивость процедуры настройки в зависимости от параметра скорости обучения.

Для обучения линейной нейронной сети может быть применена типовая процедура обучения с помощью функции **train**.

Синтаксис:

`[net,tr,Y,E,Pf,Af] = train (NET,P,T,Pi,Ai,VV,TV)`

Описание:

Тренирует сеть **NET** в соответствии с **NET.trainFcn** и **NET.trainParam**.

TRAIN(NET,P,T,Pi,Ai) в качестве входных параметров использует:
NET — сеть;

P — входы сети;

T — целевые значения, по умолчанию — нули;

Pi — начальные входные задержки, по умолчанию — нули;

Ai — начальные задержки слоев, по умолчанию — нули;

VV — структура векторов верификации, по умолчанию = `[]`;

TV — структура тестовых векторов, по умолчанию = `[]`;

и возвращает:

NET — новая сеть (тренированная);

TR — результат тренировки (количество эпох и функция выполнения);

Y — выходы сети;

E — ошибки сети;

Pf — окончательные входные задержки;

Af — окончательные задержки слоев.

Следует отметить, что T — необязательный параметр и используется только при обучении сетей, для которых необходимы эталонные значения. P_i и P_f также необязательные параметры и используются только для сетей с задержками входов и слоев. Необязательные аргументы VV и TV описаны ниже.

Аргументы **TRAIN** могут быть двух форматов: массив ячеек или матрица. Формат в виде массива ячеек является наиболее удобным для описания. Он наиболее удобен для описания сетей с многими входами и выходами и позволяет представлять последовательность входов:

P — $N_i \times TS$ массив ячеек, каждая ячейка $P\{i,ts\}$ — матрица размером $R_i \times Q$;

T — $N_t \times TS$ массив ячеек, каждая ячейка $T\{i,ts\}$ — матрица размером $V_i \times Q$;

P_i — $N_i \times ID$ массив ячеек, каждая ячейка $P_i\{i,k\}$ — матрица размером $R_i \times Q$;

A_i — $N_l \times LD$ массив ячеек, каждая ячейка $A_i\{i,k\}$ — матрица размером $S_i \times Q$;

Y — $N_O \times TS$ массив ячеек, каждая ячейка $Y\{i,ts\}$ — матрица размером $V_i \times Q$;

E — $N_t \times TS$ массив ячеек, каждая ячейка $E\{i,ts\}$ — матрица размером $V_i \times Q$;

P_f — $N_i \times ID$ массив ячеек, каждая ячейка $P_f\{i,k\}$ — матрица размером $R_i \times Q$;

A_f — $N_l \times LD$ массив ячеек, каждая ячейка $A_f\{i,k\}$ — матрица размером $S_i \times Q$,

где

$N_i = \text{net.numInputs}$

$N_l = \text{net.numLayers}$

$N_t = \text{net.numTargets}$

$ID = \text{net.numInputDelays}$

$LD = \text{net.numLayerDelays}$

$TS = \text{number of time steps}$

$Q = \text{batch size}$

$R_i = \text{net.inputs}\{i\}.\text{size}$

$S_i = \text{net.layers}\{i\}.\text{size}$

$V_i = \text{net.targets}\{i\}.\text{size}$

Столбцы P_i , P_f , A_i , and A_f упорядочены от самых больших задержек к текущим:

$P_{i\{i,k\}} = i\text{-й вход в момент времени } ts=k-ID.$

$P_{f\{i,k\}} = i\text{-й вход в момент времени } ts=TS+k-ID.$

$A_{i\{i,k\}} = i\text{-й выход слоя в момент времени } ts=k-LD.$

$A_{f\{i,k\}} = i\text{-й выход слоя в момент времени } ts=TS+k-LD.$

Матричный формат может быть использован тогда, когда должен быть смоделирован только один шаг по времени ($TS = 1$). Это удобно только для сетей с одним входом и с одним выходом. Однако он может быть использован для сетей с большим количеством входов и выходов. Каждый аргумент матрицы находится путем накопления элементов, соответствующих аргументам массивов элементов, в одну матрицу:

P — матрица (сумма R_i) $\times Q$;

T — матрица (сумма V_i) $\times Q$;

P_i — матрица (сумма R_i) $\times (ID \times Q)$;

A_i — матрица (сумма S_i) $\times (LD \times Q)$;

Y — матрица (сумма U_i) $\times Q$;

E — матрица (сумма V_i) $\times Q$;

P_f — матрица (сумма R_i) $\times (ID \times Q)$;

A_f — матрица (сумма S_i) $\times (LD \times Q)$.

Если VV и TV заданы, они должны быть пустыми матрицами или должны иметь структуру со следующими полями:

$VV.P$, $TV.P$ — входы верификация/тест;

$VV.T$, $TV.T$ — эталонные значения верификация/тест (по умолчанию — нули);

$VV.P_i$, $TV.P_i$ — начальные входные задержки верификация/тест (по умолчанию — нули);

$VV.A_i$, $TV.A_i$ — задержки слоев верификация/тест (по умолчанию — нули).

Векторы верификации используются для того, чтобы остановить тренировку раньше, если дальнейшая тренировка на исходных векторах будет ухудшать приближение к векторам верификации. Тестовый вектор функционирования может быть использован для измерения того, насколько хорошо сеть обеспечивает согласование между исходными векторами и векторами верификации. Если в качестве $VV.T$, $VV.P_i$ или $VV.A_i$ выбраны пустые матрицы или массивы элементов, то будут использованы значения по умолчанию. Это правило справедливо также и для $TV.T$, $TV.P_i$, $TV.A_i$.

Не все функции тренировки поддерживают векторы верификации и тестовые векторы. Только те, которые не игнорируют аргументы VV и TV .

Алгоритм:

Эта функция для каждого вектора входа выполняет настройку весов и смещений, используя функцию `learnp`. В результате сеть будет настраиваться по сумме всех коррекций. Каждый пересчет для набора входных векторов называется эпохой. Это и отличает процедуру обучения от процедуры адаптации `adapt`, когда настройка параметров реализуется при представлении каждого отдельного вектора входа.

Затем процедура `train` моделирует настроенную сеть для имеющегося набора векторов, сравнивает результаты с набором целевых векторов и вычисляет среднеквадратичную ошибку. Как только значение ошибки становится меньше заданного или исчерпано предельное число эпох, обучение прекращается.

Пример 6. Выполним процедуру обучения линейной сети с

$P=[1 \ -1.2]; T=[0.5 \ 1];$

```
>> clear, P = [1 -1.2];
```

```
% Вектор входов
```

```
T= [0.5, 1];
```

```
% Вектор целей
```

```
maxlr = 0.40*maxlinlr(P,'bias');
```

```
%Максимальное значение параметра обучения
```

```

net = newlin([-2,2],1,[0],maxlr);
% Создание линейной сети

w_range=-1:0.2:1; b_range=-1:0.2:1;
ES = errsurf(P,T, w_range, b_range,
'purelin');
% Расчет функции критерия качества

surfc(w_range, b_range, ES)
% Построение поверхности функции критерия ка-
чества

```

На рисунке 3.6а построена поверхность функции критерия качества в пространстве параметров сети. В процессе обучения траектория обучения будет перемещаться из начальной точки в точку минимума критерия качества.

Выполним расчет и построим траекторию обучения линейной сети для заданных начальных значений веса и смещения:

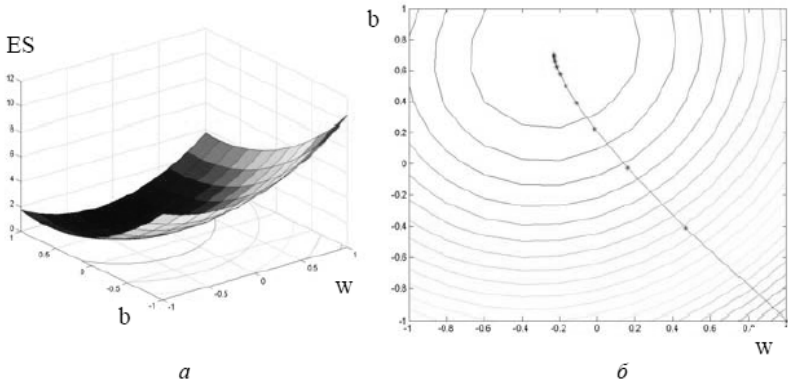


Рис. 3.6. Моделирование нейронной сети

```

>> x = zeros(1,50); y = zeros(1,50);
net.IW{1}=1; net.b{1}= -1;

```

```

x(1) = net.IW{1}; y(1) = net.b{1};
net.trainParam.goal = 0.001;
net.trainParam.epochs = 1;
% Расчет траектории обучения

for i = 2:50,
    [net, tr] = train(net,P,T)
    x(i) = net.IW{1}
    y(i) = net.b{1}
end
% Цикл вычисления весов и смещения для одной
эпохи

TRAINB, Epoch 0/1, MSE 5.245/0.001.
TRAINB, Epoch 1/1, MSE 2.049/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 2.049/0.001.
TRAINB, Epoch 1/1, MSE 0.815178/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.815178/0.001.
TRAINB, Epoch 1/1, MSE 0.330857/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.330857/0.001.
TRAINB, Epoch 1/1, MSE 0.137142/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.137142/0.001.
TRAINB, Epoch 1/1, MSE 0.0580689/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.0580689/0.001.
TRAINB, Epoch 1/1, MSE 0.0250998/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.0250998/0.001.
TRAINB, Epoch 1/1, MSE 0.0110593/0.001.
TRAINB, Maximum epoch reached.

```

```

TRAINB, Epoch 0/1, MSE 0.0110593/0.001.
TRAINB, Epoch 1/1, MSE 0.00495725/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.00495725/0.001.
TRAINB, Epoch 1/1, MSE 0.00225533/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.00225533/0.001.
TRAINB, Epoch 1/1, MSE 0.00103897/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.00103897/0.001.
TRAINB, Epoch 1/1, MSE 0.000483544/0.001.
TRAINB, Maximum epoch reached.
TRAINB, Epoch 0/1, MSE 0.000483544/0.001.
TRAINB, Performance goal met.
TRAINB, Epoch 0/1, MSE 0.000483544/0.001.
TRAINB, Performance goal met.
TRAINB, Epoch 0/1, MSE 0.000483544/0.001.
TRAINB, Performance goal met.
clear, contour(w_range, b_range, ES, 20),
hold on
plot(x, y, '-*'), grid on, hold off
% Построение линий уровня и траектории обучения

```

На рисунке 3.6б символами «*» отмечены значения веса и смещения на каждом шаге обучения; видно, что примерно за 10 шагов при заданной точности 0.001 получим $w = -0.22893$, $b = 0.70519$. Это согласуется с решением, полученным с использованием процедуры адаптации.

Если не строить траектории процесса обучения, то можно выполнить обучение, обратившись к функции `train` только один раз:

```

>> net.IW{1}=1; net.b{1}= -1;
net.trainParam.epochs = 50;

```



```

net.trainParam.goal = 0.001;
[net, tr] = train(net,P,T);
net.IW, net.b

TRAINB, Epoch 0/50, MSE 5.245/0.001.
TRAINB, Epoch 11/50, MSE 0.000483544/0.001.
TRAINB, Performance goal met.
ans =

        [-0.2289]

ans =

        [0.7052]

```

Если повысить точность обучения до значения 0.00001, то получим следующие результаты:

```

>> net.trainParam.goal = 0.00001;
[net, tr] = train(net,P,T);
net.IW, net.b

TRAINB, Epoch 0/50, MSE 0.000483544/1e-005.
TRAINB, Epoch 6/50, MSE 5.55043e-006/1e-005.
TRAINB, Performance goal met.

ans =

        [-0.2279]

ans =

        [0.7249]

```

Повышение точности на два порядка приводит к уточнению значений параметров во втором знаке.

§ 3.5. Задание для лабораторной работы

Задание 1.

1. Для заданного преподавателем варианта задания (табл. 3.1) создать линейную нейронную сеть.

Исходные данные к заданию 1

Номер варианта	Количество входов	Диапазоны значений входов	Количество нейронов в слое
1	2	-3...+3	2
2	2	-1...+1	3
3	2	-4... +4	2
4	2	-2...+2	3
5	2	-8...+8	2
6	2	-9...+9	3
7	2	-7...+7	2
8	2	-5...+5	3
9	2	-3...+3	2
10	2	-6...+6	3

2. Разработать алгоритм создания и моделирования линейной нейронной сети.

3. Реализовать разработанный алгоритм в системе MATLAB.

4. Определить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5).

5. Изменить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5).

Структурную схему нейронной сети, алгоритм, текст программы, график, выводы включить в отчет.

Задание 2.

1. Для заданного преподавателем варианта задания (табл. 3.2) сконструировать линейную сеть с помощью функции `newlind`, промоделировать ее работу и определить значения веса и смещения.

2. Построить график линий уровня поверхности функции ошибки в системе MATLAB.

Включить в отчет: структурную схему нейронной сети; алгоритм, текст программы и графики, результаты расчета ошибки в системе MATLAB; выводы.

Исходные данные к заданию 2

Номер варианта	Количество входов — 1; количество нейронов — 1			
	Значения входа персептрона		Целевой выход	
	1-е задание	2-е задание	1-е задание	2-е задание
1	{-2 1}	{-2 1 0 2}	{-1 -1}	{-1 -1 1 0}
2	{0 1}	{0 1 -1 -1}	{0 1}	{0 1 1 0}
3	{-2 1}	{-2 1 0 3}	{2 2}	{2 2 0 -2}
4	{-1 -2}	{-1 -2 1 2}	{1 -1}	{1 -1 -2 0}
5	{0 -1}	{0 -1 -1 1}	{0 1}	{0 1 0 1}
6	{-2 1}	{-2 1 3 2}	{1 -2}	{1 -2 -1 2}
7	{-2 0}	{-2 0 2 -2}	{1 1}	{1 1 -1 -1}
8	{-1 0}	{-1 0 1 1}	{-1 0}	{-1 0 -1 1}
9	{0 2}	{0 2 1 -2}	{0 -2}	{0 -2 -2 -1}
10	{-3 2}	{-3 2 2 3}	{1 -1}	{1 -1 2 -1}

Задание 3.

1. Для заданного преподавателем варианта задания (табл. 3.3) создать линейную сеть с помощью функции `newlind` и осуществить ее настройку при помощи функции `train`.

2. Построить график функции ошибки и график траектории обучения в системе MATLAB аналогично рисунку 3.6.

Включить в отчет: структурную схему нейронной сети; алгоритм, текст программы и графики; результаты расчета ошибки в системе MATLAB; выводы.

Таблица 3.3

Исходные данные к заданию 3

Номер варианта	Количество входов — 1; количество нейронов — 1		
	Диапазон значений входа	Значения входа персептрона	Целевой выход
1	-4...+4	{-2 1}	{1 -2}
2	-3...+3	{-2 0}	{1 1}
3	-2...+2	{-1 0}	{-1 0}
4	-4...+4	{0 2}	{0 -2}
5	-4...+4	{-3 2}	{1 -1}
6	-4...+4	{-2 1}	{-1 -1}
7	-2...+2	{0 1}	{0 1}
8	-4...+4	{-2 1}	{2 2}
9	-3...+3	{-1 -2}	{1 -1}
10	-2...+2	{0 -1}	{0 1}

Указание. Все решения по заданиям производить с использованием системы MATLAB. Для оформления работы использовать Microsoft WORD.

§ 3.6. Структура отчета

По материалам работы каждым студентом составляется отчет по установленной форме с использованием Microsoft WORD. Расчеты, диаграммы, графики следует выполнять с использованием ПЭВМ. Особое внимание при оформлении отчета студенты должны обратить на составление выводов по выполненной работе. В выводах нужно сопоставить результаты проведенных исследований с известными из теоретического курса закономерностями и выяснить согласованность полученных результатов с теоретическими. Полностью оформленный исполнителем отчет представляется каждым студентом преподавателю на следующем занятии.

Отчет по лабораторной работе должен содержать:

- 1) титульный лист, оформленный по образцу;
- 2) цель лабораторной работы и исходные данные в Microsoft WORD;
- 3) результаты исследований по всем заданиям в виде таблиц, рисунков, графиков, структурных схем нейронных сетей; алгоритмов, текстов программ в Microsoft WORD;
- 4) выводы по заданиям.

К отчету прилагаются программы в системе MATLAB в электронном виде.

Представленные в отчете результаты, порядок их получения, расчетов, графиков и схем студенты обязаны уметь четко пояснить. За проведенную работу и оформленный отчет преподаватель выставляет дифференцированную оценку.

Глава 4

ЛАБОРАТОРНАЯ РАБОТА «МОДЕЛИРОВАНИЕ НЕЙРОННЫХ СЕТЕЙ В MATLAB»

§ 4.1. Цель работы

Лабораторная работа выполняется на основе теоретических положений искусственного интеллекта на ПК с использованием системы MatLab (MATLAB) и Microsoft WORD в среде Windows 8/10. Цели работы:

1) изучение средств и методов MATLAB, пакетов Neural Network Toolbox и Simulink для моделирования и исследования нейронных сетей;

2) получение умений и навыков:

- в освоении базовых приемов моделирования и исследования нейронных сетей в среде MATLAB;
- в применении нейронных сетей для аппроксимации функций;
- в анализе полученных результатов.

§ 4.2. Пакет Neural Network Toolbox

Рассмотрим основные положения по нейронным сетям и возможности пакета Neural Network Toolbox на основе положений [4, 6, 7, 12–18].

Типичный пример сети с прямой передачей сигнала показан на рисунке 4.1. Нейроны регулярным образом организованы в слои. Входной слой служит просто для ввода значений входных переменных. Каждый из скрытых и выходных нейронов соединен со всеми элементами предыдущего слоя. Можно было бы рассматривать сети, в которых нейроны связаны только с некоторыми из нейронов предыдущего слоя; однако для большинства приложений сети с полной системой связей предпочтительнее.

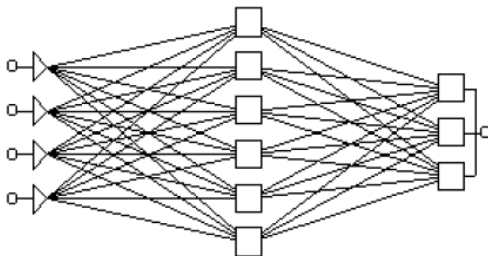


Рис. 4.1. Пример сети с прямой передачей сигнала

При работе (использовании) сети во входные элементы подаются значения входных переменных, затем сигналы последовательно обрабатывают нейроны промежуточных и выходного слоев. Каждый из них вычисляет свое значение активации, беря взвешенную сумму выходов элементов предыдущего слоя и вычитая из нее пороговое значение. Затем значение активации преобразуется с помощью функции активации, и в результате получается выход нейрона. После того как вся сеть отработает, выходные значения элементов выходного слоя принимаются за выход всей сети в целом.

Вначале сеть настраивается. Процесс настройки сети получил название «обучение сети». Перед началом обучения связям присваиваются небольшие случайные значения. Каждая итерация процедуры состоит из двух фаз. Во время первой фазы на сеть подается входной вектор путем установки в нужное состояние входных элементов. Затем входные сигналы распространяются по сети, порождая некоторый выходной вектор. Для работы алгоритма требуется, чтобы характеристика «вход – выход» нейроподобных элементов была неубывающей и имела ограниченную производную. Обычно для этого используют сигмоидную нелинейность.

Известные типы сетей: однослойный персептрон; многослойный персептрон; сеть Хэмминга; сеть Ворда; сеть Хопфилда; сеть Кохонена; когнитрон; неокогнитрон.

В состав пакета Neural Network Toolbox (Нейронные сети) входят более 160 различных функций, что позволяет пользователю создавать, обучать и использовать самые различные искусственные нейронные сети (НС). Пакет может быть использован для решения множества разнообразных задач, таких как обработка сигналов, нелинейное управление, финансовое моделирование и т. п.

Для каждого типа архитектуры и обучающего алгоритма искусственной НС (ИНС) имеются функции инициализации, обучения, адаптации, создания, моделирования, демонстрации, а также примеры применения. Искусственные многослойные нейронные сети конструируются по принципам построения их биологических аналогов. Они уже сейчас способны решать широкий круг задач распознавания образов, идентификации, управления сложными нелинейными объектами, роботами и т. п.

Представим некоторые проблемы, решаемые в контексте НС и представляющие интерес для пользователей.

Классификация образов. Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, к одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови.

Кластеризация/категоризация. При решении задачи кластеризации, которая известна также как классификация образов «без учителя», отсутствует обучающая выборка с метками классов. Алгоритм кластеризации основан на подобии образов и помещает близкие образы в один кластер. Известны случаи применения кластеризации для извлечения знаний, сжатия данных и исследования свойств данных.

Аппроксимация функций. Предположим, что имеется обучающая выборка $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$ (пары данных «вход — выход»), которая генерируется неизвестной функцией $F(x)$, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной

функции $F(x)$. Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

Предсказание/прогноз. Пусть заданы n дискретных отсчетов $(y(t_1), y(t_2), y(t_k))$ в последовательные моменты времени t_1, t_2, \dots, t_k . Задача состоит в предсказании значения $y(t_{k+1})$ в некоторый будущий момент времени t_{k+1} . Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

Оптимизация. Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Известная задача коммивояжера является классическим примером задачи оптимизации.

Память, адресуемая по содержимому. В модели вычислений фон Неймана обращение к памяти доступно только посредством адреса, который не зависит от содержимого памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержимому, доступна по указанию заданного содержимого. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержимому. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.

Управление. Рассмотрим динамическую систему, заданную совокупностью $\{u(t), y(t)\}$, где $u(t)$ является входным управляющим воздействием, а $y(t)$ — выходом системы в момент времени t . В системах управления с эталонной моделью целью управления является расчет такого входного воздействия $u(t)$, при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление двигателем.

Эти и подобные задачи успешно решаются средствами пакета Neural Networks Toolbox.

Список функций Neural Network Toolbox взят из [13, 14] и приведен в приложении.

§ 4.3. Работа с нейронной сетью в командном режиме

Рассмотрим работу с нейронной сетью в командном режиме пакета Neural Network Toolbox на основе положений [4, 7, 13, 14, 17]. Выполним исследование НС для аппроксимации выбранной функции, используя функции **newgrnn** и **newrbe** пакета Neural Network Toolbox.

Синтаксис:

`net = newgrnn (P,T,spread)`

Описание:

Обобщенные регрессионные сети являются разновидностью радиальных базисных сетей и используются для анализа временных рядов, решения задач обобщенной регрессии и аппроксимации функций. Характерной особенностью этих сетей является высокая скорость их обучения.

Функция `net = newgrnn (P, T, spread)` формирует обобщенную регрессионную сеть и имеет следующие входные и выходные аргументы.

Входные аргументы:

P — массив размера $R \times Q$ из Q входных векторов, **R** — число элементов вектора входа;

T — массив размера $S \times Q$ из Q векторов цели;

spread — параметр влияния, по умолчанию 1.0.

Выходные аргументы:

net — объект класса `network object` обобщенной регрессионной сети.

Чем больше число используемых радиальных базисных функций, тем более гладкой будет аппроксимация функции. Чтобы выполнить

точную аппроксимацию, следует использовать значение параметра `spread` меньшее, чем расстояние между векторами входа. Чтобы получить гладкую аппроксимацию, следует увеличить значение параметра `spread`.

Свойства:

Функция `newgrnn` создает двухслойную нейронную сеть, архитектура которой совпадает с архитектурой радиальной базисной сети. Первый слой включает нейроны с функцией активации `radbas` и использует функции взвешивания `dist` и накопления `netprod`. Второй, линейный слой включает нейроны с функцией активации `purelin` и использует функции взвешивания `normprod` и накопления `netsum`. Смещения используются только в первом слое.

Функция `newgrnn` устанавливает веса первого слоя равными P' , а смещения — равными $0.8326/\text{spread}$, что приводит к радиальным базисным функциям, которые пересекают величину 0.5 при значениях взвешенных входов $\pm \text{spread}$. Веса второго слоя $W2$ устанавливаются равными T .

Для выполнения работы необходимо запустить MATLAB и перейти в командное окно.

Пример 1. Создать обобщенную регрессионную сеть с входами P и целями T :

```
>> P = 0:3;
T = [0.0 2.0 4.1 5.9];
net = newgrnn(P,T);
gensim(net)
```

Выполним моделирование сети для нового входа и построим график (рис. 4.2):

```
>> plot(P,T, 'r', 'MarkerSize',2, 'LineWidth',2)
hold on
```

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

```
% Векторы входа из обучающего множества
```

```
plot(P,V,'ob','MarkerSize',8, 'LineWidth',2)
```

```
P1 = 0.5:2.5;
Y = sim(net,P1);
plot(P1,Y,'+k','MarkerSize',10,'LineWidth',2)
% Рис.4.2
```

```
Y = sim(net, 0:0.5:3)
```

```
Y = 0.8104      1.3759      2.1424      3.0300
3.9030      4.6345      5.1615
```

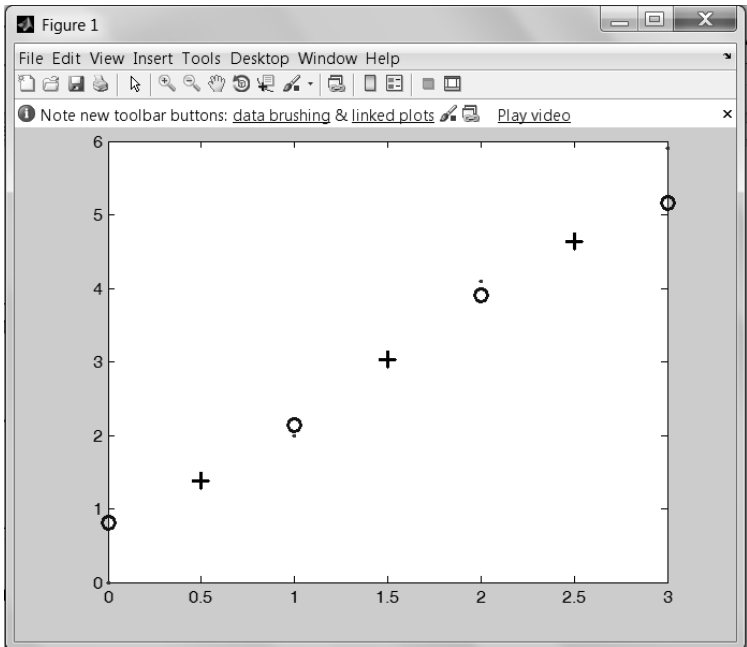


Рис. 4.2. Результат моделирования сети

Из анализа результатов моделирования следует, что на границах интервала расхождения существенны.

Если уменьшить значение параметра влияния до 0.1, то мы получим аппроксимацию высокой точности (рис. 4.3):

```
net = newgrnn(P,T,0.1);
```

```
Y = sim(net, 0:0.5:3)
```

```
Y =      0.0000      1.0000      2.0000      3.0500
4.1000      5.0000      5.9000
```

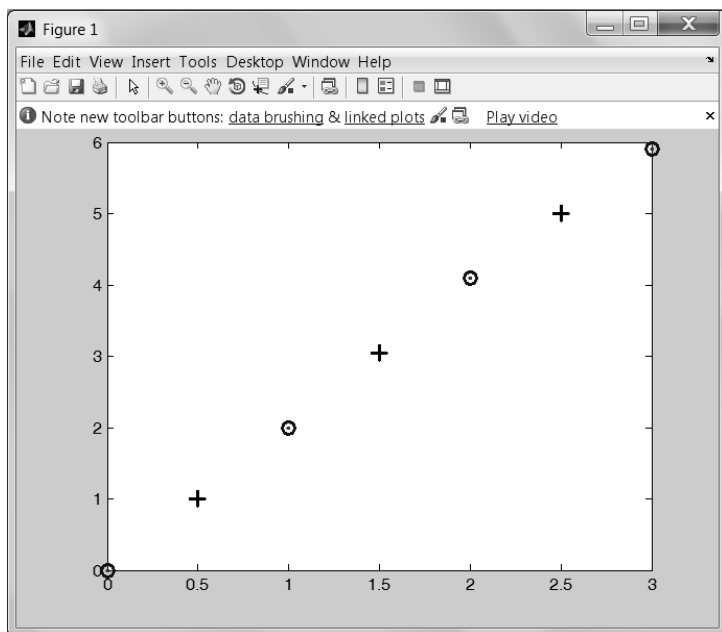


Рис. 4.3. Результат моделирования сети при уменьшении параметра влияния

Синтаксис:

```
net = newrbe (P,T,spread)
```

Описание:

Функция `net = newrb (P, T, spread)` формирует радиальную базисную сеть с нулевой ошибкой и имеет следующие входные и выходные аргументы.

Входные аргументы:

P — массив размера $R \times Q$ из Q входных векторов, R — число элементов вектора входа;

T — массив размера $S \times Q$ из Q векторов цели;

spread — параметр влияния, по умолчанию 1.0.

Выходные аргументы:

net — объект класса `network object` радиальной базисной сети с нулевой ошибкой.

Чем больше значение параметра spread , тем более гладкой будет аппроксимация. Слишком большое значение spread может привести к вычислительным проблемам.

Алгоритм:

Функция `newtgb` создает радиальную базисную сеть с двумя слоями. Первый слой включает нейроны с функцией активации `radbas` и использует функции взвешивания `dist` и накопления `netprod`. Вторым, линейным слоем включает нейроны с функцией активации `purelin` и использует функции взвешивания `dotprod` и накопления `netsum`.

Функция `newtgb` устанавливает веса первого слоя равными P' , а смещения — равными $0.8326/\text{spread}$, в результате радиальная базисная функция пересекает значение 0.5 при значениях взвешенных входов $\pm \text{spread}$. Веса второго слоя IW и смещения b определяются путем моделирования выходов первого слоя A и последующего решения системы линейных уравнений:

$$[W \ b] * [A; \text{ones}] = T$$

Пример 2. Создать радиальную базисную сеть с нулевой ошибкой для следующей обучающей последовательности:

```
>> P = 0:3;
```

```
T = [0.0 2.0 4.1 5.9];
```

```
net = newtgb(P,T);net.layers{1}.size
```

```
ans = 4
```

Сформированная радиальная базисная сеть с нулевой ошибкой имеет 4 нейрона в первом слое.

Выполним моделирование сети для нового входа (рис. 4.4).

```

>>
plot(P,T,'*r','MarkerSize',2,'LineWidth',2)
hold on
V = sim(net,P); % Векторы входа из обучающе-
го множества
plot(P,V,'ob','MarkerSize',8,'LineWidth',2)
P1 = 0.5:2.5;
Y = sim(net,P1)
plot(P1,Y,'+k','MarkerSize',10,'Lin-
ewidth',2)
Y = sim(net, 0:0.5:3)

Y = 0.0000    1.0346    2.0000    2.8817
4.1000    5.5053    5.9000

```

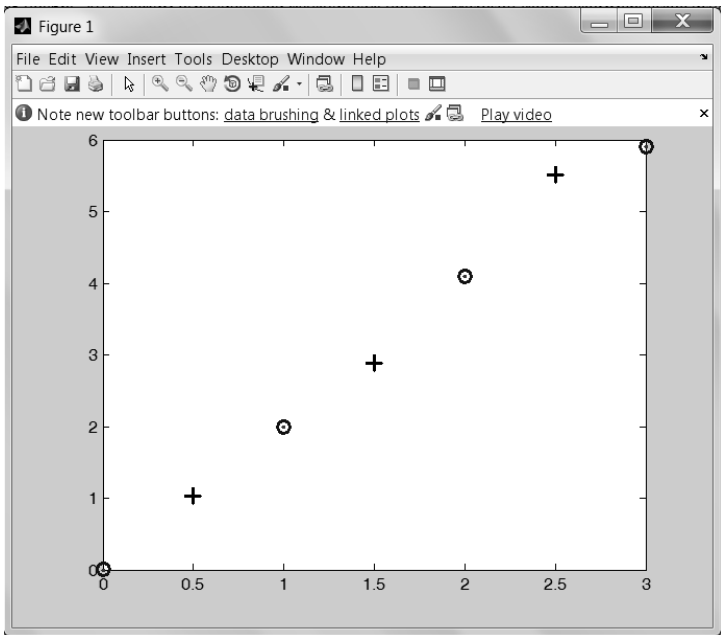


Рис. 4.4. Результат моделирования сети

Пример 3. Создать обобщенно-регрессионную НС (сеть типа GRNN) с именем *a*, реализующую функциональную зависимость между входом и выходом вида $y = x^2$ на отрезке $[-1, 1]$, используя следующие экспериментальные данные:

$x = [-1 \ -0.8 \ -0.5 \ -0.2 \ 0 \ 0.1 \ 0.3 \ 0.6 \ 0.9 \ 1],$

$y = [1 \ 0.64 \ 0.25 \ 0.04 \ 0 \ 0.01 \ 0.09 \ 0.36 \ 0.81 \ 1].$

Проверку качества восстановления приведенной зависимости осуществим, используя данные контрольной выборки $x1 = [-0.9 \ -0.7 \ -0.3 \ 0.4 \ 0.8]$, которым соответствуют значения $y1 = [0.81 \ 0.49 \ 0.09 \ 0.16 \ 0.64]$.

Решение. Процедура создания и использования данной НС описывается следующим образом:

```
>> x = [-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9
1];
```

```
>> % Задание входных значений
```

```
>> y = [1 0.64 0.25 0.04 0 0.01 0.09 0.36
0.81 1];
```

```
>> % Задание целевых значений
```

```
>> a=newgrnn(x,y,0.01); % Создание НС с от-
клонением 0.01
```

```
>> Y1 = sim(a,[-0.9 -0.7 -0.3 0.4 0.8]) %
Опрос НС
```

```
Y1 = 0.8200      0.5049      0.0316      0.0710
0.6390
```

Как видно, точность аппроксимации в данном случае получилась не очень высокой — максимальная относительная погрешность аппроксимации составляет 30,61%.

Можно попытаться улучшить качество аппроксимации за счет подбора величины отклонения, но в условиях примера приемлемый результат легко достигается при использовании сети с радиальными базисными элементами типа *newrbe*:

```
>> a=newrbe(x,y);
```

```
>>Y1 = sim(a,[-0.9 -0.7 -0.3 0.4 0.8])
```

‡ Опрос НС

Y1 = 0.8100 0.4900 0.0900 0.1600 0.6400

Нетрудно видеть, что применение сети типа `newrb` приводит здесь не просто к интерполяции заданных обучающей выборкой значений, а, действительно, к точному восстановлению заданной зависимости — по крайней мере, для использованных точек контрольной последовательности.

Созданную сеть можно сохранить для последующего использования набором в командной строке команды ***save ('a')***; при этом будет создан файл ***a.mat***, т. е. файл с именем НС и расширением ***mat***. В последующих сеансах работы эту сеть можно загрузить, используя функцию ***load ('a')***. Естественно, допустимы все другие формы записи операторов ***save*** и ***load***.

Отметим, что в условиях примеров дать какую-либо оценку предельной величине погрешности аппроксимации невозможно, особенно для значений входов, выходящих за пределы диапазона входов обучающей последовательности. К сожалению, это является характерной особенностью нейросетевых моделей. Для подавляющего числа задач, решаемых с помощью аппарата нейронных сетей, не только для задач классификации или прогноза, каких-либо вероятностных оценок точности получаемых решений получить не удается.

§ 4.4. Использование GUI-интерфейса пакета нейронных сетей

Для создания НС в Fuzzy Logic Toolbox имеется графический интерфейс пользователя (GUI, или ГИП). Рассмотрим порядок работы с ним на основе положений [4, 7, 13, 14, 17].

4.4.1. Создание нейронной сети

Пример 4. Создать, используя графический интерфейс пользователя, нейронную сеть для выполнения операции $y = x^2$ при задании векторов входа

$x = [-1 \ -0.8 \ -0.5 \ -0.2 \ 0 \ 0.1 \ 0.3 \ 0.6 \ 0.9 \ 1]$ и цели

$y = [1 \ 0.64 \ 0.25 \ 0.04 \ 0 \ 0.01 \ 0.09 \ 0.36 \ 0.81 \ 1]$.

Создадим векторы входа и целей:

```
>> x = [-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9
1];
>> y = [1 0.64 0.25 0.04 0 0.01 0.09 0.36
0.81 1];
```

Для открытия основного окна интерфейса необходимо в командном окне MATLAB ввести команду

```
>> nntool
```

Выполнение команды приведет к открытию окна создания нейронной сети **Network/Data Manager** (рис. 4.5).

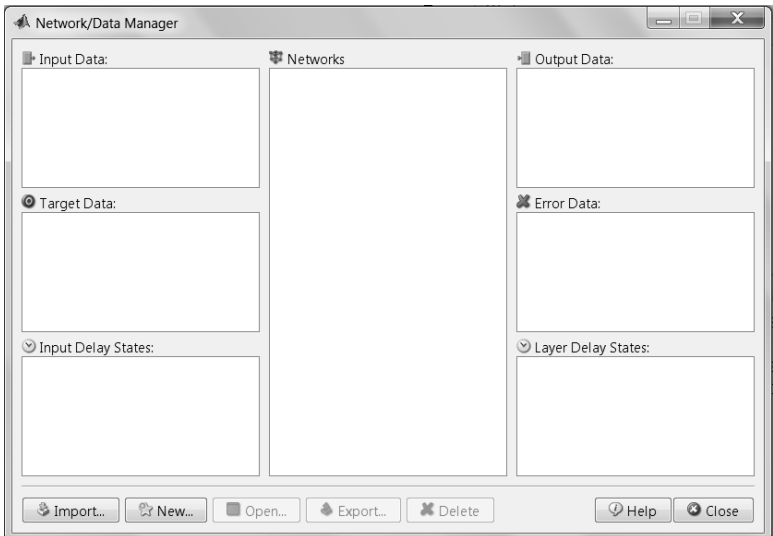


Рис. 4.5. Окно создания нейронной сети Network/Data Manager

Используя кнопку **Import**, откроем окно **Import to Network/Data Manager** (рис. 4.6), выберем вектор входа **x** в качестве входных данных **Input Data** и нажмем кнопку **Import**.

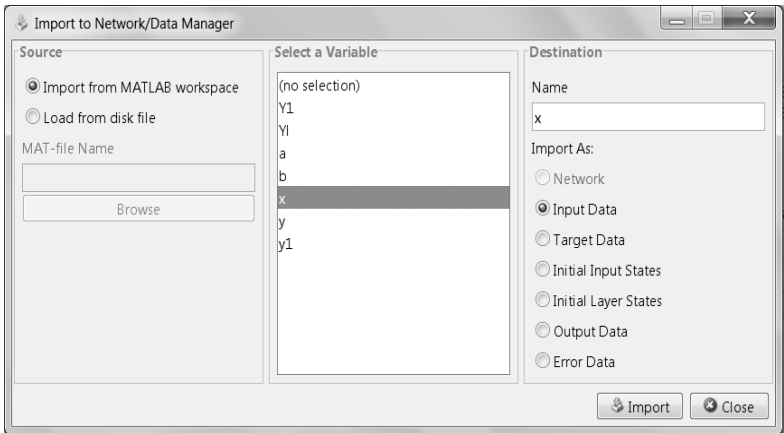


Рис. 4.6. Окно импорта в нейронную сеть
Import to Network/Data Manager

В результате появится окно **Imported** (рис. 4.7). Аналогичную операцию проделаем для вектора целей **y** и целевых данных **Target Data**.

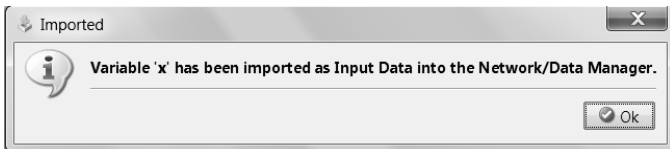


Рис. 4.7. Окно импорта в нейронную сеть Imported

Закроем окно **Import to Network/Data Manager** кнопкой **Close**.

Создадим новую нейронную сеть. Для этого в окне **Network/Data Manager** нажмем кнопку **New**.

В открывшемся окне **Create Network or Data** выберем нейронную сеть типа **feed-forward backprop** с прямой передачей сигнала и с об-

ратным распространением ошибки. При создании сети сохраним ей имя, даваемое по умолчанию (**network1**) (рис. 4.8).

В качестве входных данных **Input Data** выберем **x**, а в качестве целевых данных **Target Data** — **y**. Количество нейронов (**Number of neurons**) первого слоя (**Layer 1**) установим равным двум.

Остальные установки при создании сети оставим по умолчанию.

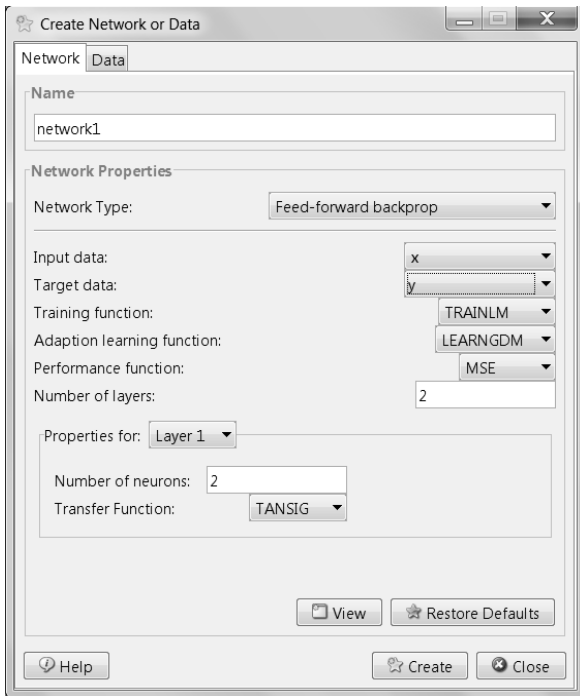


Рис. 4.8. Окно создания новой нейронной сети
Create Network or Data

Создание сети завершим нажатием кнопки **Create**.

После этого в окне **Network/Data Manager**, в области **Network**, появится имя новой созданной сети — **network1** (рис. 4.9). Дважды щелкнем по этому имени левой кнопкой мыши, что приведет к открытию окна **Network: network1** (рис. 4.10).

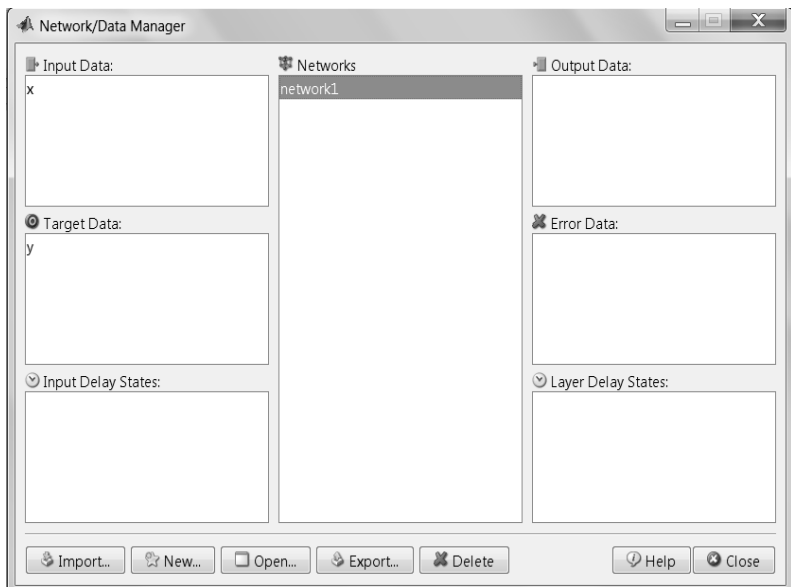


Рис. 4.9. Окно создания нейронной сети
Network/Data Manager

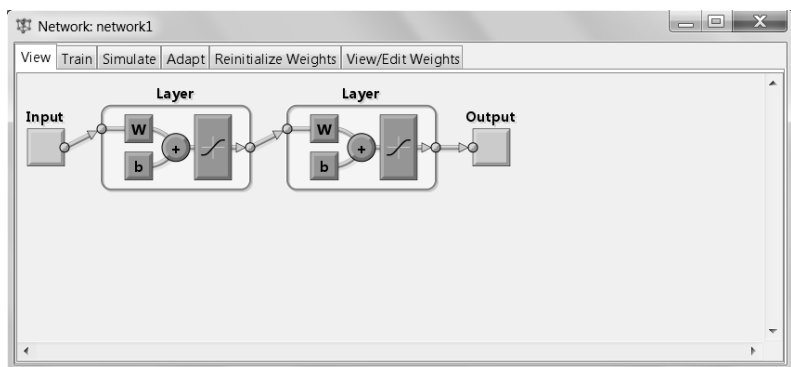


Рис. 4.10. Окно созданной нейронной сети
Network: network1

Для ввода в панели **Network: network1** установленных диапазонов и инициализации весов можно воспользоваться вкладкой **Reinitialize Weights**. Если требуется вернуться к прежним диапазонам, то следует выбрать кнопки **Revert Input Ranges** (Вернуть диапазоны) и **Revert Weights** (Вернуть веса) (рис. 4.11).

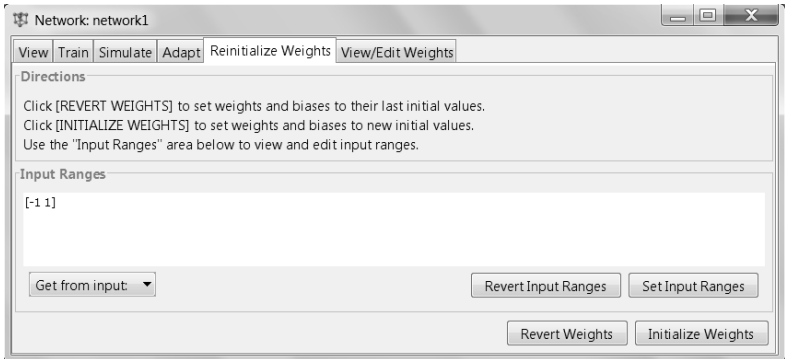


Рис. 4.11. Вкладка **Reinitialize Weights** окна созданной нейронной сети **Network: network1**

4.4.2. Обучение нейронной сети

Для обучения созданной сети выбирается вкладка **Train** в панели **Network: network1** и открывается новая диалоговая панель.

Панель имеет две вкладки:

Training Info (Информация об обучающих последовательностях);

Training Parameters (Параметры обучения).

Пример 5. Применяя эти вкладки, можно установить имена последовательностей входа и цели (на вкладке **Training Info** — в левой ее части необходимо указать x и y), а также значения параметров процедуры обучения (на вкладке **Training Parameters**; в условиях примера сохраним значения по умолчанию) (рис. 4.12).

Для обучения созданной сети нажмем кнопку **Train Network**, в результате чего откроется окно **Neural Network Training** (рис. 4.13). Качество обучения сети на выбранной обучающей последовательности отображается графиком (рис. 4.14).

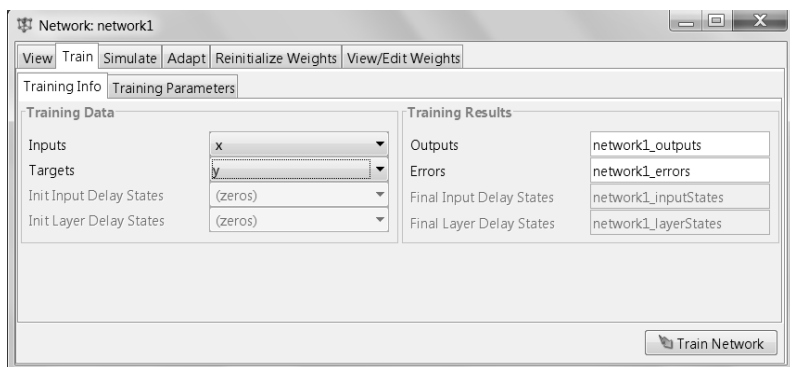


Рис. 4.12. Вкладка Training Info окна Train созданной нейронной сети Network: network1

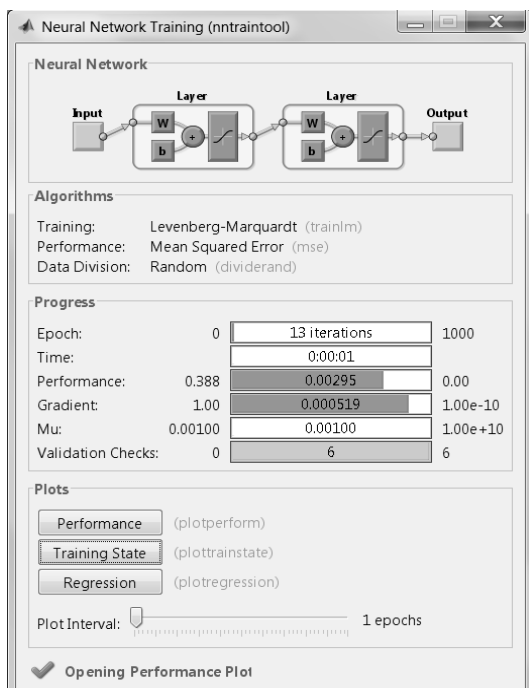


Рис. 4.13. Окно Neural Network Training созданной нейронной сети Network: network1

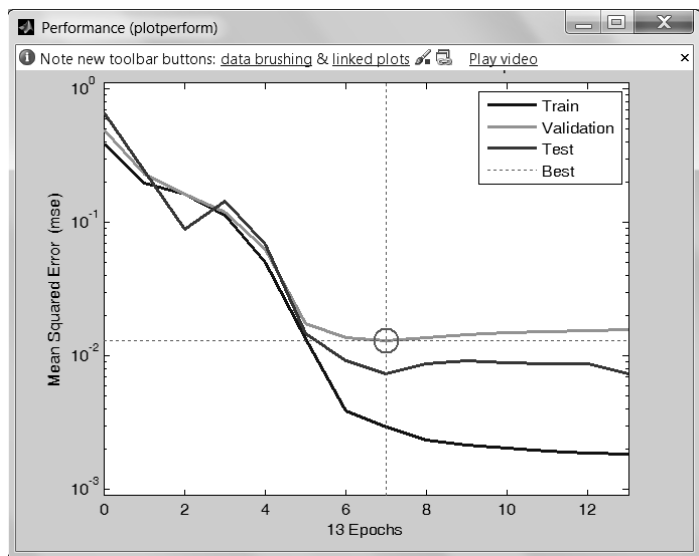


Рис. 4.14. Окно Performance созданной нейронной сети Network: network1

График либо выводится в виде отдельного окна самостоятельно, либо, в более поздней версии MATLAB, вызывается нажатием кнопки **Performance** в окне **Neural Network Training**. Видно, что к концу процесса обучения ошибка становится очень малой (вид данного рисунка при повторе вычислений может отличаться от приведенного).

Результаты обучения можно просмотреть в окне **Network/Data Manager**, активизируя имена последовательностей выходов **network1_outputs** или ошибок **network1_errors** двойным щелчком левой кнопки мыши (рис. 4.15).

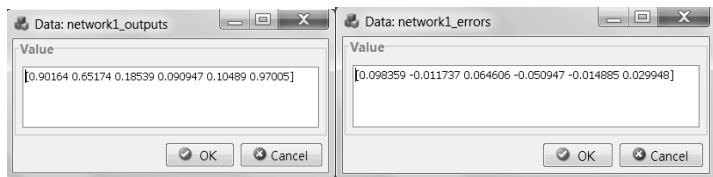


Рис. 4.15. Окна network1_outputs и network1_errors созданной нейронной сети Network: network1

Следует отметить, что в данном случае точность аппроксимации заданной функции получилась не очень высокой — максимальная абсолютная погрешность составляет 0.098, относительная — 9,8%, в чем можно убедиться, просмотрев значения ошибок (**network1_errors**) или выходов (**network1_outputs**) сети.

Заметим, что точность аппроксимации здесь можно было бы повысить, конструируя сеть с большим числом нейронов, но при этом необходима и более представительная обучающая выборка.

4.4.3. Работа с созданной сетью

Пример 6. Для просмотра структурной схемы сети необходимо, выбрав имя сети (**network1**), дважды щелкнуть по этому имени левой кнопкой мыши, что приведет к открытию окна **Network: network1** (см. рис. 4.12).

При необходимости можно экспортировать созданную нейронную сеть в рабочую область системы MATLAB, нажав кнопку **Export** (в открывшемся окне необходимо выбрать название НС, а затем нажать кнопку **Export**), и получить информацию о весах и смещениях непосредственно в рабочем окне системы, выполнив команду:

```
>> network1.IW{1,1},network1.b{1}
```

```
ans =  
    3.5561  
    2.5895
```

```
ans =  
   -2.4207  
    2.0983
```

и команду:

```
>> network1.IW{2,1},network1.b{2}
```

```
ans =
```


[]

ans =

2,0288

Основной функцией для формирования нейросетевых моделей в Simulink является функция **gensim**, записываемая в форме **gensim(net,st)**, где **net** — имя созданной НС, **st** — интервал дискретизации (если НС не имеет задержек, ассоциированных с ее входами или выходами, значение данного аргумента устанавливается равным -1).

Теперь можно построить модель НС в среде Simulink и отобразить ее схему, используя команду:

```
>> gensim(network1)
```

Эта схема (рис. 4.16) является в полной мере функциональной схемой и может быть применена для моделирования нейронной сети.

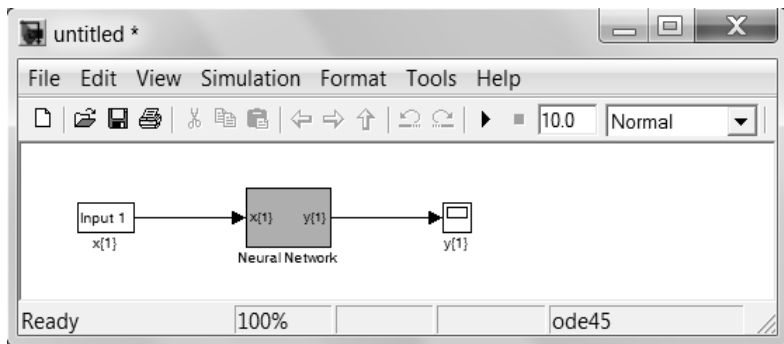


Рис. 4.16. Функциональная схема НС network1

Двойной щелчок на блоке **Neural Network** раскрывает слои сети, а двойной щелчок на блоке слоя сети раскрывает его структуру.

При выполнении команды **gensim** кроме схемы НС открывается окно **Library:neural** с набором блоков, которые можно использовать для внесения изменений в схему (рис. 4.17).

Пакет **Neural Network Toolbox** содержит ряд блоков, которые могут быть либо непосредственно использованы для построения нейронных сетей в среде **Simulink**, либо применяться вместе с рассмотренной выше функцией **gensim**.

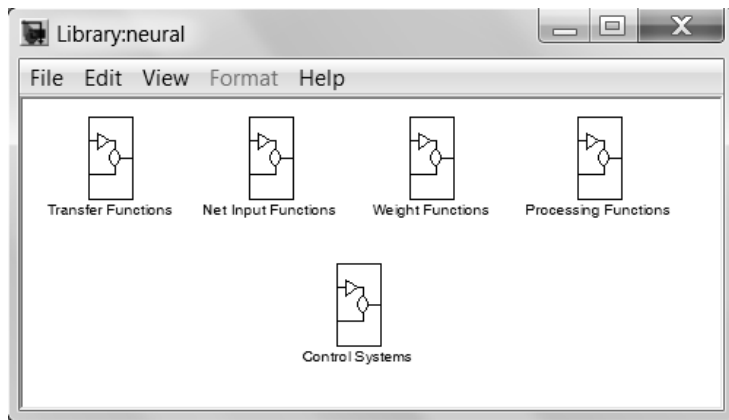


Рис. 4.17. Библиотека блоков Library:neural

Для вызова этого набора блоков в командной строке MATLAB необходимо набрать команду **neural**, после выполнения которой появляется окно Library:neural (этого же результата можно добиться с помощью кнопки **Simulink** меню MATLAB и далее — кнопки **Neural Network Toolbox**) (рис. 4.18).

Каждый из нейросетевых блоков, в свою очередь, является набором (библиотекой) некоторых блоков. Рассмотрим их.

Двойной щелчок на блоке **Transfer Functions** приводит к появлению библиотеки блоков функций активации. Каждый из блоков данной библиотеки преобразует подаваемый на него вектор в соответствующий вектор той же размерности.

Блок **Net Input Functions** включает блоки, реализующие функции преобразования входов сети.

Блок **Weight Functions** содержит библиотеку блоков, реализующих некоторые функции весов и смещений.

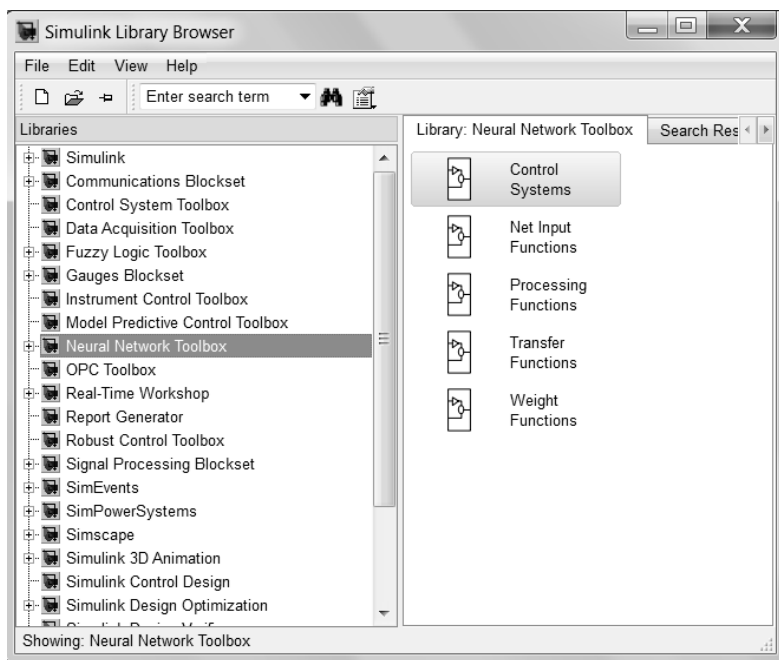


Рис. 4.18. Библиотека блоков Simulink Library Browser

Отметим, что в процессе работы со всеми приведенными блоками при задании конкретных числовых значений ввиду особенностей **Simulink** векторы необходимо представлять как столбцы, а не как строки.

Блоки, объединенные в библиотеку **Control Systems**, реализуют нейросетевые регуляторы трех различных структур — регулятор с предсказанием, регулятор, основанный на использовании модели нелинейной авторегрессии со скользящим средним (**Nonlinear Autoregressive-Moving Average — NAR-MA-L2**), и регулятор на основе эталонной модели, которые удобны при построении и исследовании моделей систем автоматического управления, а также блок просмотра результатов.

§ 4.5. Моделирование нейронных сетей при помощи Simulink

Рассмотрим моделирование нейронных сетей при помощи Simulink на основе положений [4, 7, 12–14, 17].

4.5.1. Средства Simulink для работы с нейронными сетями

Программа Simulink является приложением к пакету MATLAB. При моделировании с использованием Simulink реализуется принцип визуального программирования, в соответствии с которым пользователь на экране из библиотеки стандартных блоков создает модель устройства и осуществляет расчеты. При этом, в отличие от классических способов моделирования, пользователю не нужно досконально изучать язык программирования и численные методы математики, а достаточно общих знаний, требующихся при работе на компьютере, и, естественно, знаний той предметной области, в которой он работает. Simulink является достаточно самостоятельным инструментом MATLAB, и при работе с ним совсем не требуется знать сам MATLAB и остальные его приложения. С другой стороны, доступ к функциям MATLAB и другим его инструментам остается открытым, и их можно использовать в Simulink. Часть входящих в состав пакетов имеет инструменты, встраиваемые в Simulink (например, LTI-Viewer приложения Control System Toolbox — пакета для разработки систем управления). Имеются также дополнительные библиотеки блоков для разных областей применения (например, Power System Blockset — моделирование электротехнических устройств, Digital Signal Processing Blockset — набор блоков для разработки цифровых устройств и т. д.).

При работе с Simulink пользователь имеет возможность модернизировать библиотечные блоки, создавать свои собственные, а также составлять новые библиотеки блоков.

При моделировании пользователь может выбирать метод решения дифференциальных уравнений, а также способ изменения модельного времени (с фиксированным или переменным шагом). В ходе мо-

делирования имеется возможность следить за процессами, происходящими в системе. Для этого используются специальные устройства наблюдения, входящие в состав библиотеки Simulink. Результаты моделирования могут быть представлены в виде графиков или таблиц.

Преимущество Simulink заключается также в том, что она позволяет пополнять библиотеки блоков с помощью подпрограмм, написанных как на языке MATLAB, так и на языках C++, Fortran и Ada.

4.5.2. Обзорщик разделов библиотеки Simulink

Окно обзорщика библиотеки блоков содержит следующие элементы (рис. 4.19):

- заголовок, с названием окна — **Simulink Library Browser**;
- меню, с командами **File, Edit, View, Help**;
- панель инструментов с ярлыками наиболее часто используемых команд;
- окно комментария для вывода поясняющего сообщения о выбранном блоке;
- список разделов библиотеки, реализованный в виде дерева;
- окно содержимого раздела библиотеки (список вложенных разделов библиотеки или блоков);
- строку состояния, содержащую подсказку по выполняемому действию.

Библиотека Simulink содержит следующие основные разделы:

- **Continuous** — линейные блоки;
- **Discrete** — дискретные блоки;
- **Functions & Tables** — функции и таблицы;
- **Math** — блоки математических операций;
- **Nonlinear** — нелинейные блоки;
- **Signals & Systems** — сигналы и системы;
- **Sinks** — регистрирующие устройства;
- **Sources** — источники сигналов и воздействий;
- **Subsystems** — блоки подсистем.

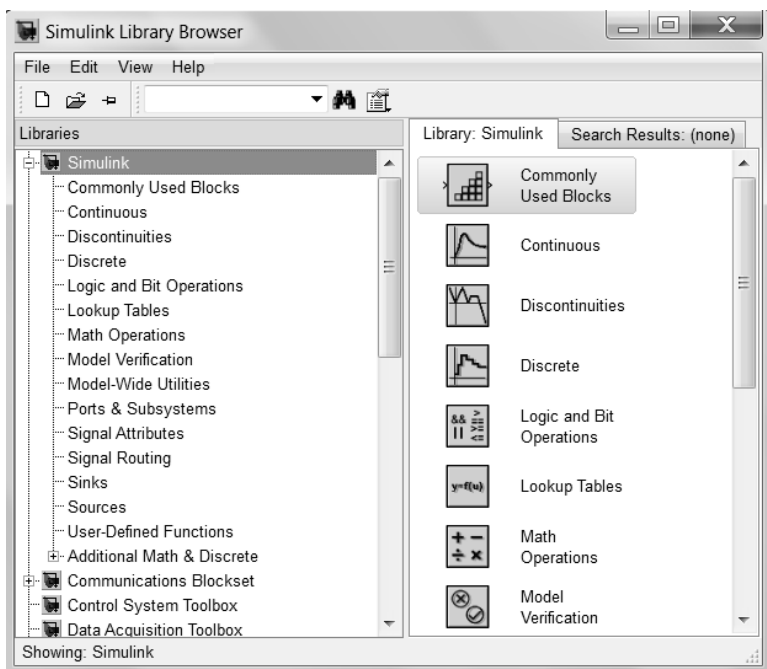



Рис. 4.19. Библиотека блоков Simulink Library Browser

Список разделов библиотеки Simulink представлен в виде дерева, и правила работы с ним являются общими для списков такого вида. Пиктограмма свернутого узла дерева содержит символ «+», а пиктограмма развернутого содержит символ «-». Для того чтобы развернуть или свернуть узел дерева, достаточно щелкнуть на его пиктограмме левой клавишей мыши.

При выборе соответствующего раздела библиотеки в правой части окна отображается его содержимое.

4.5.3. Создание модели

Для создания модели в среде Simulink необходимо создать новый файл модели с помощью команды File/New Model или используя

кнопку  на панели инструментов (здесь и далее с помощью символа «/» указаны пункты меню программы, которые необходимо последовательно выбрать для выполнения указанного действия). Вновь созданное окно модели показано на рисунке 4.20.

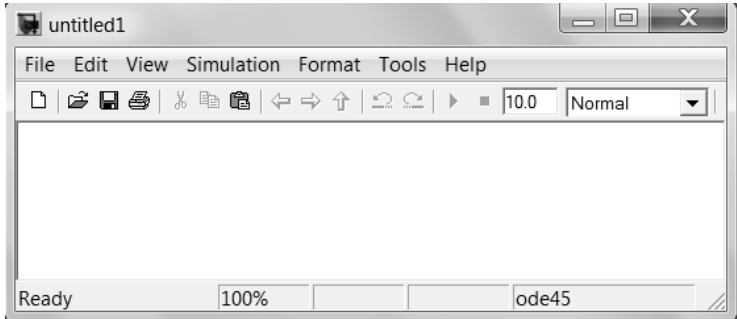


Рис. 4.20. Пустое окно модели

Затем можно располагать блоки в окне модели. Для этого необходимо открыть соответствующий раздел библиотеки (например, Sources — Источники).

Далее, указав курсором на требуемый блок и нажав на левую клавишу мыши, перетащить блок в созданное окно. Клавишу мыши нужно держать нажатой.

Для удаления блока необходимо выбрать блок (указать курсором на его изображение и нажать левую клавишу мыши), а затем нажать клавишу Delete на клавиатуре.

Для изменения размеров блока требуется выбрать блок, установить курсор в один из углов блока и, нажав левую клавишу мыши, изменить размер блока (курсор при этом превратится в двухстороннюю стрелку).

Для изменения параметров блока, установленных программой по умолчанию, необходимо дважды щелкнуть левой клавишей мыши, указав курсором на изображение блока. Откроется окно редактирования параметров данного блока. При задании численных параметров следует иметь в виду, что в качестве десятичного разделителя должна

использоваться точка, а не запятая. После внесения изменений нужно закрыть окно кнопкой ОК.

После установки на схеме всех блоков из требуемых библиотек нужно выполнить соединение элементов схемы. Для соединения блоков необходимо указать курсором на «выход» блока, а затем нажать и, не отпуская левую клавишу мыши, провести линию к входу другого блока. После чего отпустить клавишу. В случае правильного соединения изображение стрелки на входе блока изменяет цвет.

Для создания точки разветвления в соединительной линии нужно подвести курсор к предполагаемому узлу и, нажав правую клавишу мыши, протянуть линию. Для удаления линии требуется выбрать линию (так же, как это выполняется для блока), а затем нажать клавишу Delete на клавиатуре.

После составления расчетной схемы необходимо сохранить ее в виде файла на диске, выбрав пункт меню File/Save As... в окне схемы и указав папку и имя файла.

Следует иметь в виду, что имя файла не должно превышать 32 символов, должно начинаться с буквы и не может содержать символы кириллицы и спецсимволы. Это же требование относится и к пути файла (к тем папкам, в которых сохраняется файл).

При последующем редактировании схемы можно пользоваться пунктом меню File/Save. При повторных запусках программы Simulink загрузка схемы осуществляется с помощью меню File/Open... в окне обозревателя библиотеки или из основного окна MATLAB.

4.5.4. Окно модели

Окно модели (см. рис. 4.20) содержит следующие элементы:

- заголовок с названием окна. Вновь созданному окну присваивается имя Untitled с соответствующим номером;
- меню с командами **File, Edit, View** и т. д.;
- панель инструментов;
- окно для создания схемы модели;

- строку состояния, содержащую информацию о текущем состоянии модели.

Меню окна содержит команды для редактирования модели, ее настройки и управления процессом расчета, работы с файлами и т. п.:

- **File (Файл)** — работа с файлами моделей;
- **Edit (Редактирование)** — изменение модели и поиск блоков;
- **View (Вид)** — управление показом элементов интерфейса;
- **Simulation (Моделирование)** — задание настроек для моделирования и управление процессом расчета;
- **Format (Форматирование)** — изменение внешнего вида блоков и модели в целом;
- **Tools (Инструментальные средства)** — применение специальных средств для работы с моделью (отладчик, линейный анализ и т. п.);
- **Help (Справка)** — вывод окон справочной системы.

Для работы с моделью можно также использовать кнопки на панели инструментов (рис. 4.21).

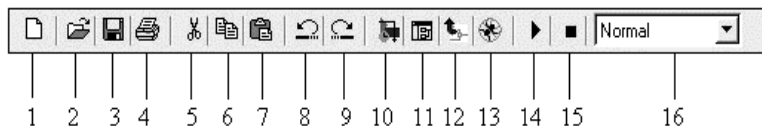


Рис. 4.21. Панель инструментов окна модели

Кнопки панели инструментов имеют следующее назначение:

- 1) **New Model** — открыть новое (пустое) окно модели;
- 2) **Open Model** — открыть существующий **mdl**-файл;
- 3) **Save Model** — сохранить **mdl**-файл на диске;
- 4) **Print Model** — вывод на печать блок-диаграммы модели;
- 5) **Cut** — вырезать выделенную часть модели в буфер промежуточного хранения;
- 6) **Copy** — скопировать выделенную часть модели в буфер промежуточного хранения;

7) **Paste** — вставить в окно модели содержимое буфера промежуточного хранения;

8) **Undo** — отменить предыдущую операцию редактирования;

9) **Redo** — восстановить результат отмененной операции редактирования;

10) **Library Browser** — открыть окно обозревателя библиотек;

11) **Toggle Model Browser** — открыть окно обозревателя модели;

12) **Go to parent system** — переход из подсистемы в систему высшего уровня иерархии («родительскую систему»). Команда доступна, только если открыта подсистема;

13) **Debug** — запуск отладчика модели;

14) **Start/Pause/Continue Simulation** — запуск модели на исполнение (команда **Start**); после запуска модели на изображении кнопки выводится символ **II**, и ей соответствует уже команда **Pause** (Приостановить моделирование); для возобновления моделирования следует щелкнуть по той же кнопке, поскольку в режиме паузы ей соответствует команда **Continue** (Продолжить);

15) **Stop** — закончить моделирование. Кнопка становится доступной после начала моделирования, а также после выполнения команды **Pause**;

16) **Normal/Accelerator** — обычный/ускоренный режим расчета. Инструмент доступен, если установлено приложение **Simulink Performance Tool**.

В нижней части окна модели находится строка состояния, в которой отображаются краткие комментарии к кнопкам панели инструментов, а также к пунктам меню, когда указатель мыши находится над соответствующим элементом интерфейса. Это же текстовое поле используется и для индикации состояния Simulink: **Ready** (Готов) или **Running** (Выполнение). В строке состояния отображаются также:

- масштаб отображения блок-диаграммы (в процентах, исходное значение равно 100%);
- индикатор степени завершенности сеанса моделирования (появляется после запуска модели);

- текущее значения модельного времени (выводится также только после запуска модели);
- используемый алгоритм расчета состояний модели (метод решения).

4.5.5. Выполнение моделирования

В качестве примера использования средств Simulink рассмотрим следующую задачу.

Пример 7. Пусть входной и целевой векторы имеют вид:

$p = [1 \ 2 \ 3 \ 4 \ 5]$; $t = [1 \ 3 \ 5 \ 7 \ 9]$;

Создадим линейную НС и протестируем ее по данным обучающей выборки:

```
>> p = [1 2 3 4 5];
>> t = [1 3 5 7 9];
>> net = newlind(p,t);
>> Y = sim(net,p)
```

$Y = \quad 1.0000 \quad 3.0000 \quad 5.0000 \quad 7.0000 \quad 9.0000$

Затем запустим Simulink командой:

```
>> gensim(net, -1)
```

Это приведет к открытию окна с нейросетевой моделью (рис. 4.22).

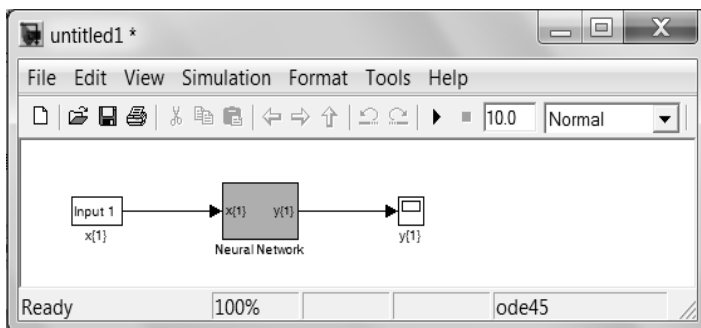


Рис. 4.22. Функциональная схема НС

Для проведения тестирования модели щелчком дважды на левом значке (с надписью **Input 1** — Вход 1), что приведет к открытию диалогового окна параметров блока. В данном случае блок **Input 1** является стандартным блоком задания константы (**Constant**). Изменим значение по умолчанию на **2** и нажмем кнопку **OK**.

Затем нажмем кнопку **Start** в панели инструментов окна модели.

Расчет нового значения сетью производится практически мгновенно. Для его вывода необходимо дважды щелкнуть на правом значке (на блоке **y(1)**).

Результат вычислений равен **3**, как и требуется, и выводится в виде графика. Для вывода результата в числовом виде к выходу модели следует подключить блок «**дисплей**», который находится в библиотеке блоков Simulink Library Browser в разделе Sinks (рис. 4.23).

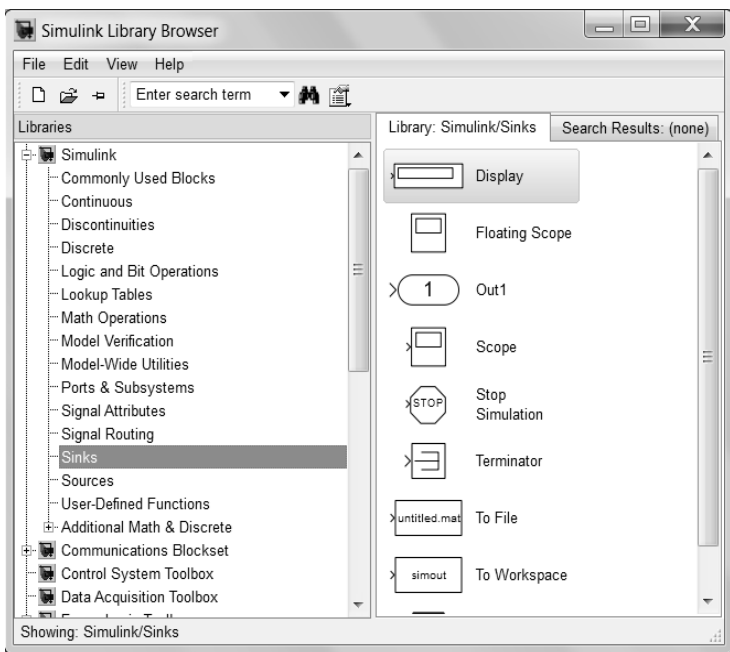


Рис. 4.23. Библиотека блоков Simulink Library Browser

Отметим, что, дважды щелкая на блоке **Neural Network**, а затем на блоке **Layer 1**, можно получить детальную графическую информацию о структуре сети (рис. 4.24).

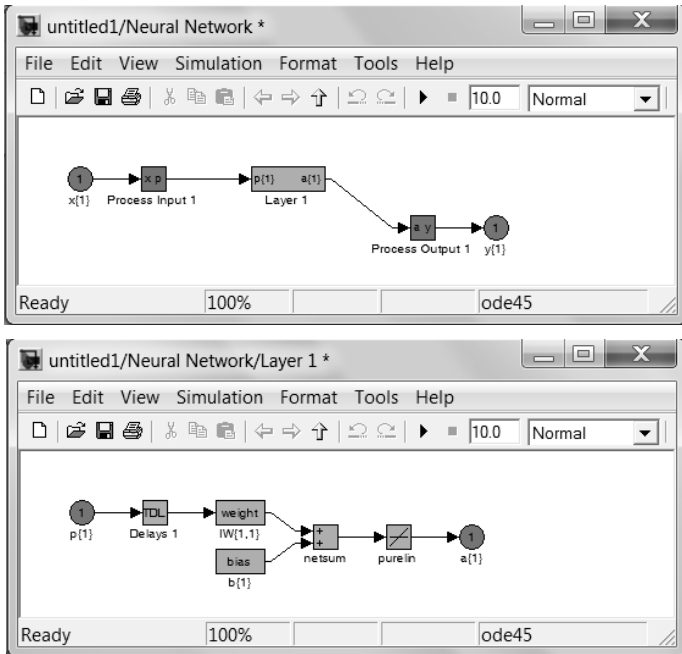


Рис. 4.24. Функциональная схема НС

С созданной сетью можно проводить различные эксперименты, возможные в среде Simulink; вообще, с помощью команды **gensim** осуществляется интеграция созданных нейросетей в блок-диаграммы этого пакета с использованием имеющихся при этом инструментов моделирования различных систем (например, возможно встраивание нейросетевого регулятора в систему управления и моделирование последней и т. п.).

4.5.6. Разработка НС для решения систем обыкновенных дифференциальных уравнений в системе Simulink

Искусственные нейронные сети широко используются для решения как инженерных, так и научных задач. Поскольку они оказались весьма эффективным средством обработки информации, постоянно делаются попытки расширить область их применения или найти новые принципы их построения и работы.

Рассмотрим задачу решения систем обыкновенных дифференциальных уравнений (ОДУ) в нейросетевом базисе. Нейросетевой базис представляет собой операцию нелинейного преобразования взвешенной суммы, т. е. операцию взвешенного суммирования нескольких входных сигналов с последующим преобразованием этой суммы посредством функции активации формального нейрона, которая может быть как линейной, так и нелинейной.

Задана система n дифференциальных уравнений 1-го порядка:

$$\dot{Y}(x) = A Y(x), \quad (4.1)$$

где A — квадратная матрица постоянных коэффициентов размера $n \times n$; Y — n -мерный вектор искомой функции аргумента x .

Для метода Рунге — Кутты 1-го порядка (метод прямоугольников) решение системы (4.1) можно представить в виде:

$$Y_{t+1} = Y_t + h \cdot A \cdot Y_t = (E + h \cdot A) Y_t = B_t \cdot Y_t, \quad (4.2)$$

где E — единичная матрица размером $n \times n$.

Элементы матрицы B есть первые два члена разложения матричной экспоненты в степенной ряд.

Для системы двух уравнений выражение (4.2) принимает вид:

$$\begin{aligned} y_{1t+1} &= b_{11}y_{1t} + b_{12}y_{2t} = (1 + ha_{11})y_{1t} + ha_{12}y_{2t}, \\ y_{2t+1} &= b_{21}y_{1t} + b_{22}y_{2t} = ha_{21}y_{1t} + (1 + ha_{22})y_{2t}. \end{aligned}$$

Схема соединения нейронов, реализующая решение системы, показана на рисунке 4.25.

Здесь $Ne1$ и $Ne2$ — нейроны, участвующие в операции интегрирования. Выходом сети являются сигналы y_1 и y_2 . Входные сигналы g_{10} и g_{20} вводят в нейроны начальное возбуждение, эквивалентное

начальным условиям решения системы уравнений. Функция активации у обоих нейронов — симметричная линейная функция.

Пример 8. Разработать НС для решения системы уравнений:

$$y'1 = y2,$$

$$y'2 = -y1$$

при начальных условиях: $y1(0) = -1, y2(0) = 0, t = 0 \dots 2\pi$.

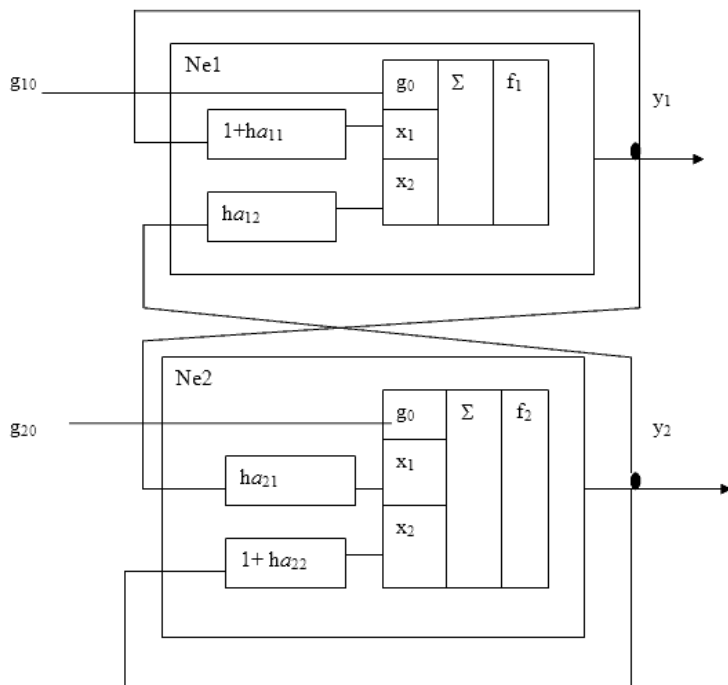


Рис. 4.25. Схема соединения нейронов, реализующая решение системы ОДУ в нейросетевом базисе методом Рунге — Кутты 1-го порядка

Блок-схема решения системы уравнений в нейросетевом базисе, созданная в среде Simulink, показана на рисунке 4.26.

Для решения задачи необходимо запустить Simulink командой:

```
>> simulink
```

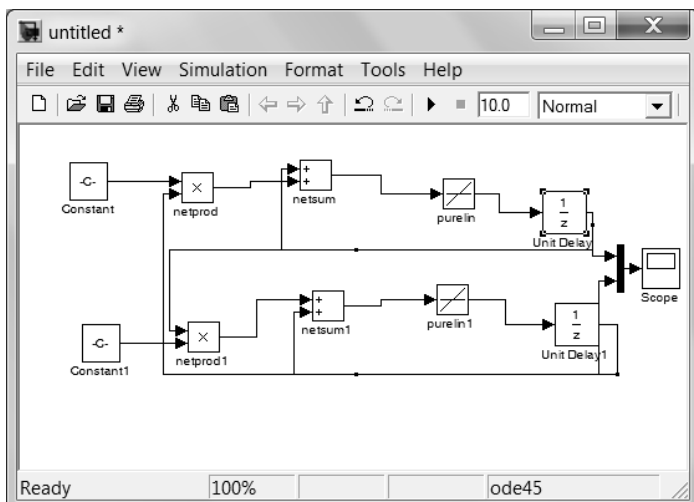


Рис. 4.26. Блок-схема решения системы уравнений в нейросетевом базисе

В открывшемся окне Simulink Library Browser нажмите кнопку New Model.

При построении схемы для задания шага интегрирования используются блоки **Constant** из раздела **Sources** (из меню **Simulink**) (рис. 4.27).

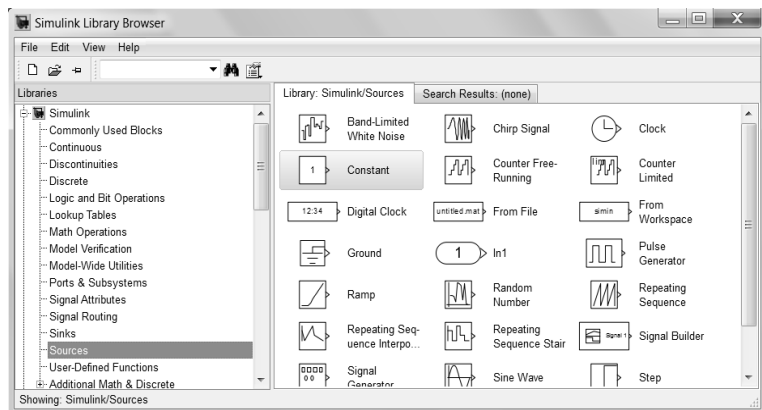


Рис. 4.27. Библиотека блоков Simulink: Sources

Вытащите два таких блока (**Constant** и **Constant1**), разместите в пустом окне модели и измените значения констант на $2\pi/100$ и $-2\pi/100$ (рис. 4.28).

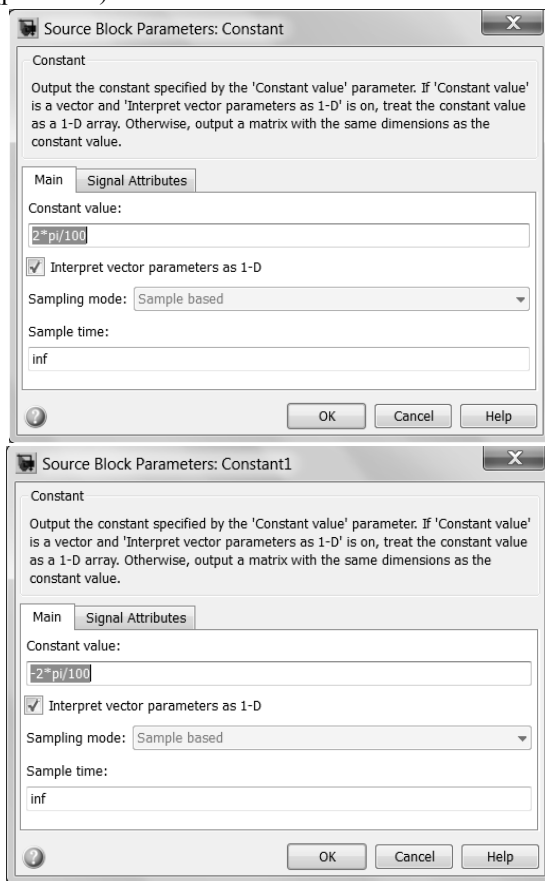


Рис. 4.28. Окно задания параметров блока Constant

Затем соедините выходы блоков **Constant** и **Constant1** с двумя блоками **netprod** из раздела **Net Input Functions** (из меню **Neural Network Toolbox**) (рис. 4.29), причем выход **Constant** соединяется с верхним входом блока **netprod**, а **Constant1** — с нижним входом

netprod1. Выходы блоков **netprod** и **netprod1** соедините с двумя блоками **netsum** из раздела **Net Input Functions** (из меню **Neural Network Toolbox**) (см. рис. 4.29), причем выход **netprod** соединяется с нижним входом блока **netsum**, а выход **netprod1** — с верхним входом **netsum1**.

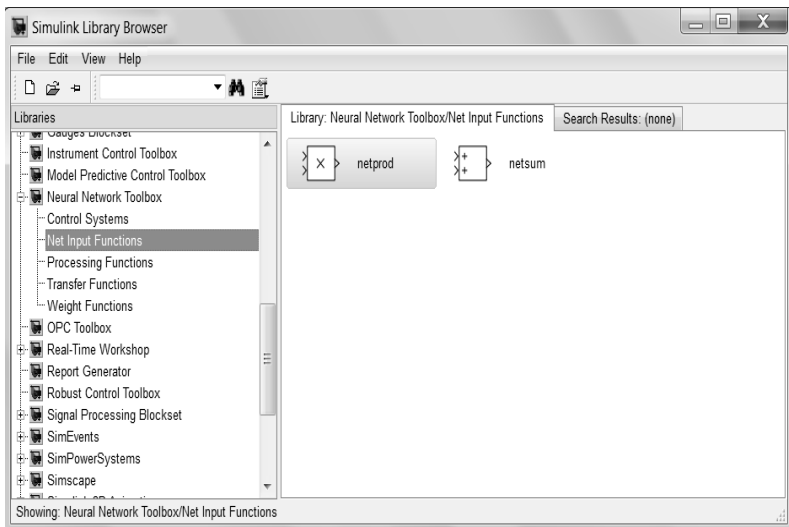


Рис. 4.29. Библиотека блоков Neural Network Toolbox:
Net Input Functions

Вытащите два блока **purelin** из раздела **Transfer Functions** (из меню **Neural Network Toolbox**) и соедините их входы с выходами блоков **netsum** и **netsum1** (рис. 4.30).

Выходы блоков **purelin** и **purelin1** соедините с входами блоков **Unit Delay** из раздела **Discrete** (из меню **Simulink**), использующихся для установки начальных значений (рис. 4.31). Измените значения **Sample Time** обоих блоков на $2\pi/100$ (рис. 4.32).

Соедините выход **Unit Delay** с верхними входами блоков **netprod1** и **netsum**, а выход **Unit Delay1** — с нижними входами **netprod** и **netsum1**.

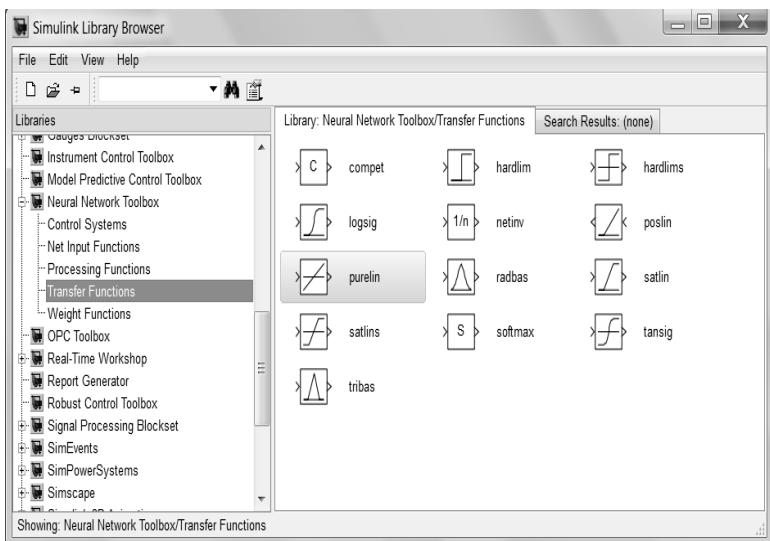


Рис. 4.30. Библиотека блоков Neural Network Toolbox:
Transfer Functions

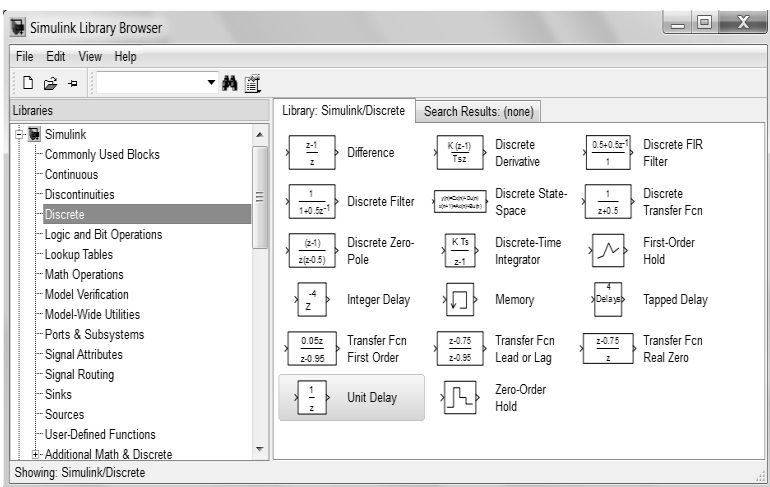


Рис. 4.31. Библиотека блоков
Simulink: Discrete

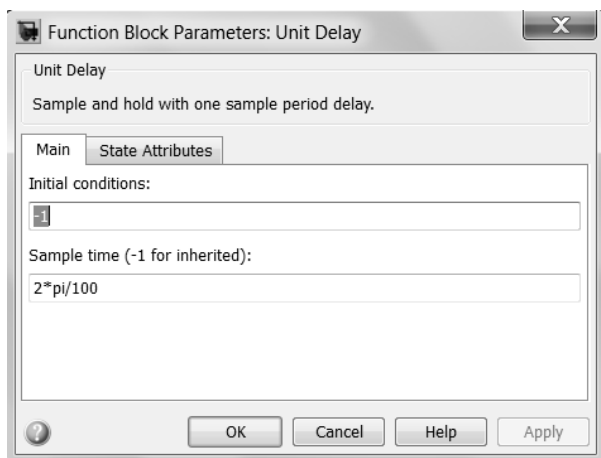
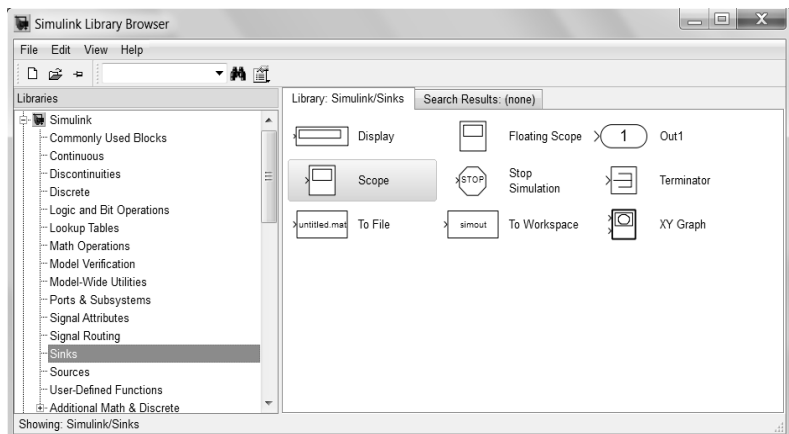


Рис. 4.32. Окно задания параметров блока Unit Delay

Для визуализации результатов используется блок **Scope** из раздела **Sinks** (из меню Simulink) (рис. 4.33).

Рис. 4.33. Библиотека блоков
Simulink: Sinks

Для показа двух графиков в блоке **Scope** вытащите блок **Mux** из раздела **Signal Routing** (из меню Simulink) и соедините его входы с выходами блоков **Unit Delay** и **Unit Delay1**, а выход — с входом блока **Scope** (рис. 4.34).

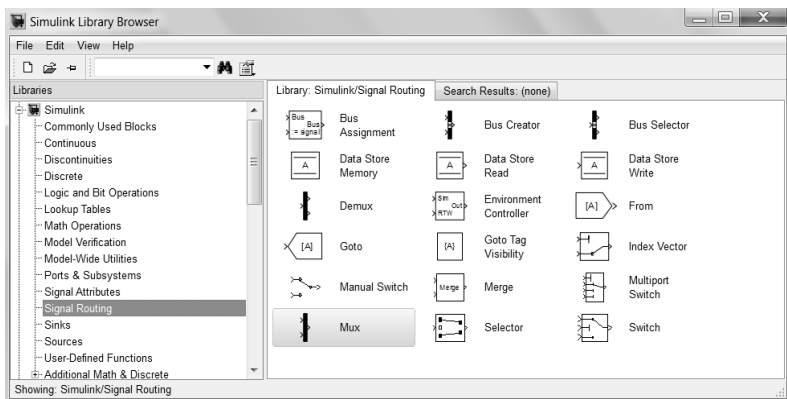


Рис. 4.34. Библиотека блоков
Simulink: Signal Routing

После того как схема собрана, можно нажимать кнопку **Start**. Решение выводится в виде графика (рис. 4.35).

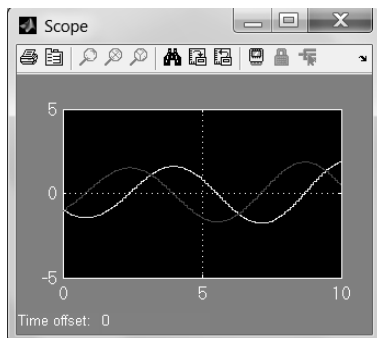


Рис. 4.35. Графики решения задачи
в Simulink

§ 4.6. Задание для лабораторной работы

Задание 1.

1. Создать обобщенно-регрессионную НС и радиальную базисную сеть с нулевой ошибкой, для чего выполнить действия, указанные в п. 4.3 в **примерах 1, 2**.

2. Восстановить зависимость по имеющимся экспериментальным данным с использованием НС, для чего выполнить действия, указанные в п. 4.3 в **примере 3**. Результаты внести в отчет в виде картинок изображений, полученных в MATLAB. Созданную сеть сохранить.

Задание 2.

1. Создать НС в Fuzzy Logic Toolbox с помощью графического интерфейса пользователя (GUI, или ГИП) для выполнения операции $y = x^2$ по исходным данным и указаниям п. 4.4.1 (см. **пример 4**).

2. Провести обучение нейронной сети и проверить ее работу по п. 4.4.2 и п. 4.4.3 (см. **примеры 5, 6**).

3. Вывести структурную схему сети и получить информацию о весах и смещениях непосредственно в рабочем окне системы.

Результаты внести в отчет в виде картинок изображений, рисунков, схем, полученных в MATLAB. Результаты сохранить в электронном виде.

Задание 3.

Провести моделирование нейронных сетей при помощи Simulink по п. 4.5.5. Исходные данные взять из **примера 7**. Результаты внести в отчет в виде картинок изображений, рисунков, схем, полученных в MATLAB. Результаты сохранить в электронном виде.

Задание 4.

Разработать НС для решения системы ОДУ (**пример 8**). Результаты внести в отчет в виде картинок изображений, рисунков, схем, полученных в MATLAB. Результаты сохранить в электронном виде.

Указание. Все решения по заданиям производить с использованием системы MATLAB. Для оформления работы использовать Microsoft WORD.

§ 4.7. Структура отчета

По материалам работы каждым студентом составляется отчет по установленной форме с использованием Microsoft WORD. Расчеты, диаграммы, графики следует выполнять с использованием ПЭВМ. Особое внимание при оформлении отчета студенты должны обратить на составление выводов по выполненной работе. В выводах нужно сопоставить результаты проведенных исследований с известными из теоретического курса закономерностями и выяснить согласованность полученных результатов с теоретическими. Полностью оформленный исполнителем отчет представляется каждым студентом преподавателю на следующем занятии.

Отчет по лабораторной работе должен содержать:

- 1) титульный лист, оформленный по образцу;
- 2) цель лабораторной работы и исходные данные в Microsoft WORD;
- 3) результаты исследований по всем заданиям в виде таблиц, рисунков, графиков, структурных схем нейронных сетей; алгоритмов, текстов программ в Microsoft WORD;
- 4) выводы по заданиям.

К отчету прилагаются программы в системе MATLAB в электронном виде.

Представленные в отчете результаты, порядок их получения, расчетов, графиков и схем студенты обязаны уметь четко пояснить. За проведенную работу и оформленный отчет преподаватель выставляет дифференцированную оценку.

Глава 5

ЛАБОРАТОРНАЯ РАБОТА

«МОДЕЛИРОВАНИЕ И РЕАЛИЗАЦИЯ НЕЙРО-НЕЧЕТКОЙ СЕТИ В СРЕДЕ MATLAB»

§ 5.1. Цель работы

Лабораторная работа выполняется на основе теоретических положений искусственного интеллекта на ПК с использованием систем MatLab (MATLAB), Microsoft WORD и EXCEL в среде Windows 8/10.

Цели работы:

1) изучение методов моделирования и принципов функционирования нейро-нечетких сетей с использованием средств и методов MATLAB;

2) получение умений и навыков:

- в конструировании нейро-нечетких сетей в среде MATLAB;
- в анализе полученных результатов.

§ 5.2. Нечеткая сеть TSK

Рассмотрим методы моделирования и принципы функционирования нейро-нечетких сетей с использованием средств и методов MATLAB на основе положений [4, 6, 7, 12–14, 17, 20].

Первоначально рассмотрим нечеткую нейронную сеть Такаги — Сугено — Канга (TSK). Нечеткая сеть TSK содержит только два параметрических слоя (первый и третий), параметры которых уточняются в процессе обучения (рис. 5.1). Параметры первого слоя называют нелинейными параметрами, поскольку они относятся к нелинейной функции, а параметры третьего слоя — линейными весами, так как они относятся к параметрам r_{kj} линейной функции TSK. При наличии N входных переменных каждое правило формирует $N+1$ переменных $r_j^{(k)}$ линейной зависимости TSK. При M правилах вывода это дает $M(N+1)$ линейных параметров сети. В свою очередь, каждая функция принадлежности использует три параметра (c , s , b), подде-

жащих адаптации. Если принять, что каждая переменная x_i характеризуется собственной функцией принадлежности (ФП), то при M правилах вывода мы получим $3MN$ нелинейных параметров. В сумме это дает $M(4N+1)$ линейных и нелинейных параметров, значения которых должны подбираться в процессе обучения сети. Обобщенную схему вывода модели TSK при использовании M правил и N переменных x_j можно представить в виде

$$\text{IF } (x_1.\text{IS}.A_1^{(1)}).\text{AND}.(x_2.\text{IS}.A_2^{(1)}).\text{AND}.\dots.\text{AND}.(x_n.\text{IS}.A_n^{(1)}),$$

$$\text{THEN } y_1 = p_{10} + \sum_{j=1}^N p_{1j}x_j$$

...

$$\text{IF } (x_1.\text{IS}.A_1^{(M)}).\text{AND}.(x_2.\text{IS}.A_2^{(M)}).\text{AND}.\dots.\text{AND}.(x_n.\text{IS}.A_n^{(M)}),$$

$$\text{THEN } y_M = p_{M0} + \sum_{j=1}^N p_{Mj}x_j.$$

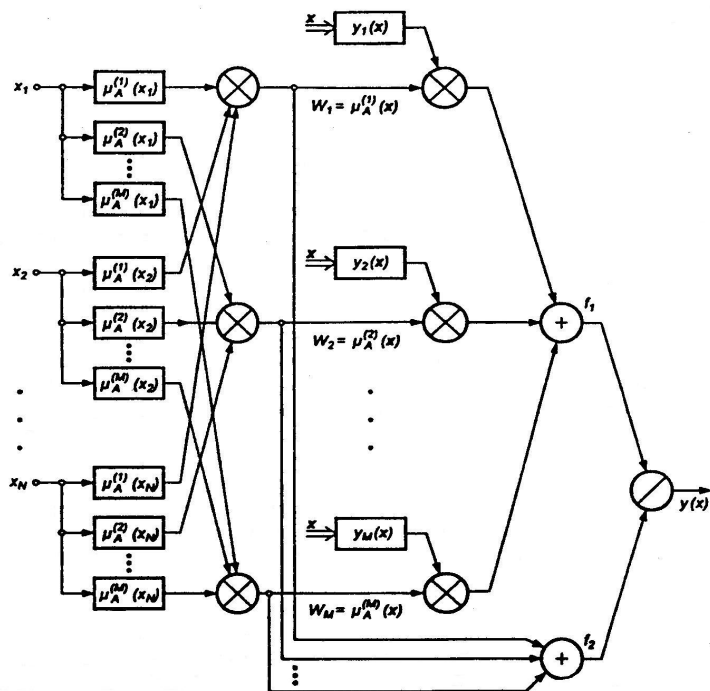


Рис. 5.1. Структура нечеткой нейронной сети TSK

Условие IF (x_i .IS. A_i) реализуется функцией фаззификации, которая представляется обобщенной функцией Гаусса отдельно для каждой переменной x_i :

$$\mu_A(x_i) = \frac{1}{1 + \left(\frac{x_i - c_i}{\sigma_i} \right)^{2b_i}}, \quad (5.1)$$

где $\mu_A(x_i)$ представляет оператор A_i . В нечетких сетях целесообразно задавать это условие в форме алгебраического произведения, из которого следует, что для k -го правила вывода

$$\mu_A^{(k)}(x) = \prod_{j=1}^N \left[\frac{1}{1 + \left(\frac{x_j - c_j^{(k)}}{\sigma_j^{(k)}} \right)^{2b_j^{(k)}}} \right]. \quad (5.2)$$

При M правилах вывода агрегирование выходного результата сети производится по формуле

$$y = \frac{\sum_{i=1}^M w_i}{\sum_{j=1}^N w_j} \left(p_{i0} + \sum_{j=1}^N p_{ij} x_j \right), \quad (5.3)$$

которую можно представить в виде

$$y(x) = \frac{1}{\sum_{k=1}^N w_k} \sum_{k=1}^M w_k y_k(x), \quad (5.4)$$

где $y_k(x) = p_{k0} + \sum_{j=1}^N p_{kj} x_j$. Присутствующие в этом выражении веса

w_k интерпретируются как значимость компонентов $\mu_A^{(k)}(x)$, определенных формулой (5.1). При этом условии формулу (5.4) можно сопоставить с многослойной структурой сети (см. рис. 5.1).

На практике для уменьшения количества адаптируемых параметров оперируют меньшим количеством независимых функций принадлежности для отдельных переменных, руководствуясь правилами, в которых комбинируются функции принадлежности различных переменных. Если принять, что каждая переменная x_i имеет m различных функций принадлежности, то максимальное количество правил, которые можно создать при их комбинировании, составит: $M = m^N$ (при трех функциях принадлежности, распространяющихся на две переменные, это $3^2 = 9$ правил вывода). Таким образом, суммарное количество нелинейных параметров сети при M правилах вывода уменьшается с $3MN$ в общем случае до $3NM^{1/N}$. Количество линейных параметров при подобной модификации остается без изменений, т. е. $M(N+1)$.

§ 5.3. Гибридная сеть как адаптивная система нейро-нечеткого вывода

Гибридная сеть представляет собой многослойную нейронную сеть специальной структуры без обратных связей, в которой используются обычные (не нечеткие) сигналы, веса и функции активации, а выполнение операции суммирования основано на использовании фиксированной Т-нормы, Т-конормы или некоторой другой непрерывной операции. При этом значения входов, выходов и весов гибридной нейронной сети представляют собой вещественные числа из отрезка $[0, 1]$ [4, 7, 17].

Основная идея, положенная в основу модели гибридных сетей, заключается в том, чтобы использовать существующую выборку данных для определения параметров функций принадлежности, которые лучше всего соответствуют некоторой системе нечеткого вывода. При этом для нахождения параметров функций принадлежности используются известные процедуры обучения нейронных сетей.

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме так называемой адаптивной системы нейро-нечеткого вывода ANFIS (редактор гибридных систем). С одной сто-

роны, гибридная сеть ANFIS представляет собой нейронную сеть с единственным выходом и несколькими входами, которые представляют собой нечеткие лингвистические переменные. При этом термы входных лингвистических переменных описываются стандартными для системы MATLAB функциями принадлежности, а термы выходной переменной представляются линейной или постоянной функцией принадлежности.

С другой стороны, гибридная сеть ANFIS представляет собой систему нечеткого вывода FIS типа Сугено нулевого или первого порядка, в которой каждое из правил нечетких продукций имеет постоянный вес, равный 1. Как известно, FIS Editor, или FIS-редактор, — это редактор нечеткой системы вывода Fuzzy Inference System Editor вместе со вспомогательными программами — редактором функции принадлежности (Membership Function Editor), редактором правил (Rule Editor), просмотрщиком правил (Rule Viewer) и просмотрщиком поверхности отклика (Surface Viewer).

§ 5.4. Моделирование и реализация нейро-нечеткой сети в среде MATLAB

Рассмотрим принципы функционирования нейро-нечетких сетей с использованием средств и методов MATLAB на основе положений [4, 7, 14, 17].

В пакете Fuzzy Logic Toolbox системы MATLAB гибридные сети реализованы в форме адаптивных систем нейро-нечеткого вывода ANFIS. При этом разработка и исследование гибридных сетей оказываются возможными:

- в интерактивном режиме с помощью специального графического редактора адаптивных сетей, получившего название редактора ANFIS;
- в режиме командной строки с помощью ввода имен соответствующих функций с необходимыми аргументами непосредственно в окно команд системы MATLAB.

Редактор ANFIS позволяет создавать или загружать конкретную модель адаптивной системы нейро-нечеткого вывода, выполнять ее обучение, визуализировать ее структуру, изменять и настраивать ее параметры, а также использовать настроенную сеть для получения результатов нечеткого вывода.

5.4.1. Описание ANFIS-редактора

ANFIS является аббревиатурой Adaptive Neuro-Fuzzy Inference System — адаптивной нейро-нечеткой системы. ANFIS-редактор позволяет автоматически синтезировать из экспериментальных данных нейро-нечеткие сети. Нейро-нечеткую сеть можно рассматривать как одну из разновидностей систем нечеткого логического вывода типа Сугено. При этом функции принадлежности синтезированных систем настроены (обучены) так, чтобы минимизировать отклонения между результатами нечеткого моделирования и экспериментальными данными. Загрузка ANFIS-редактора осуществляется по команде `anfisedit`. В результате выполнения этой команды появится графическое окно (рис. 5.2).

На этом же рисунке указаны функциональные области ANFIS-редактора, описание которых приведено ниже.

ANFIS-редактор содержит (см. рис. 5.2):

- 3 верхних меню — **File**, **Edit** и **View**;
- область визуализации;
- область свойств ANFIS;
- область загрузки данных;
- область генерирования исходной системы нечеткого логического вывода;
- область обучения, область тестирования;
- область вывода текущей информации;
- а также кнопки **Help** и **Close**, которые позволяют вызвать окно справки и закрыть ANFIS-редактор соответственно.

Меню **File** (рис. 5.3) и **View** (рис. 5.4) одинаковые для всех GUI-модулей, используемых с системами нечеткого логического вывода.

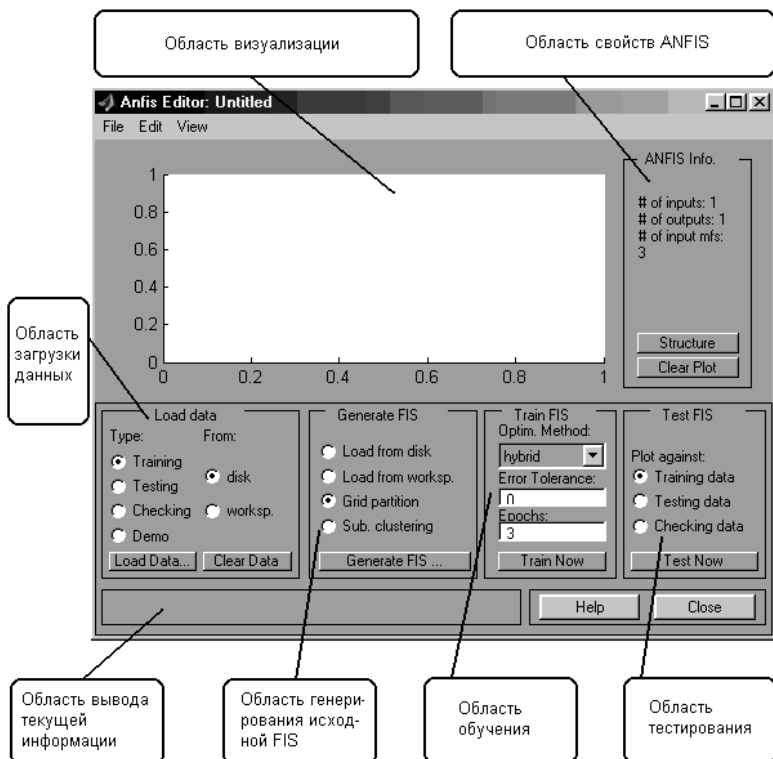


Рис. 5.2 Основное окно ANFIS-редактора

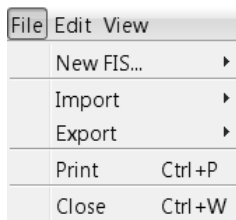


Рис. 5.3. Меню File

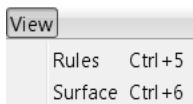


Рис. 5.4. Меню View

Общий вид меню **Edit** приведен на рисунке 5.5.

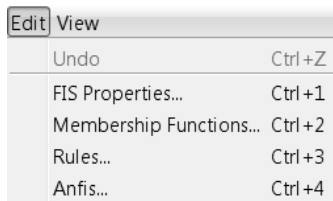


Рис. 5.5. Меню Edit

Команда **Undo** отменяет ранее совершенное действие. Выполняется также по нажатию **Ctrl+Z**.

Команда **FIS Properties...** открывает FIS-редактор. Эта команда может быть также выполнена нажатием **Ctrl+1**.

Команда **Membership Functions...** открывает редактор функций принадлежности. Эта команда может быть также выполнена нажатием **Ctrl+2**.

Команда **Rules...** открывает редактор базы знаний. Эта команда может быть также выполнена нажатием **Ctrl+3**.

Команда **Anfis...** открывает ANFIS-редактор. Эта команда может быть также выполнена нажатием **Ctrl+3**. Заметим, что данная команда, запущенная из ANFIS-редактора, не приводит к выполнению каких-либо действий, так как этот редактор уже открыт. Однако в меню **Edit** других GUI-модулей, используемых с системами нечеткого логического вывода, добавляется команда **Anfis...**, позволяющая открыть ANFIS-редактор из этих модулей.

Область визуализации

В этой области (см. рис. 5.2) выводится два типа информации:

- при обучении системы — кривая обучения в виде графика зависимости ошибки обучения от порядкового номера итерации;

- при загрузке данных и тестировании системы — экспериментальные данные и результаты моделирования.

Экспериментальные данные и результаты моделирования выводятся в виде множества точек в двумерном пространстве. При этом по оси абсцисс откладывается порядковый номер строчки данных в выборке (обучающей, тестирующей или контрольной), а по оси ординат — значение выходной переменной для данной строчки выборки. Используются следующие маркеры:

- голубая точка (.) — тестирующая выборка;
- голубая окружность (o) — обучающая выборка;
- голубой плюс (+) — контрольная выборка;
- красная звездочка (*) — результаты моделирования.

Область свойств ANFIS

В области свойств ANFIS (**ANFIS Info**) (см. рис. 5.2) выводится информация о количестве входных и выходных переменных, о количестве функций принадлежности для каждой входной переменной, а также о количестве строчек в выборках. В этой области расположены две кнопки: **Structure** и **Clear Plot**.

Нажатие кнопки **Structure** открывает новое графическое окно (рис. 5.6), в котором система нечеткого логического вывода представляется в виде нейро-нечеткой сети. В качестве иллюстрации приведена нейро-нечеткая сеть, содержащая четыре входных переменных и одну выходную. В этой системе по три лингвистических термина используется для оценки каждой из входных переменных и четыре термина для выходной.

Нажатие кнопки **Clear Plot** позволяет очистить область визуализации.

Область загрузки данных

В области загрузки данных (**Load Data**) (см. рис. 5.2) расположены:

1) меню выбора типа данных (**Type**), содержащее альтернативы:

- **Traning** — обучающая выборка;
- **Testing** — тестирующая выборка;
- **Checking** — контрольная выборка;
- **Demo** — демонстрационный пример;

2) меню выбора источника данных (**From**), содержащее альтернативы:

- **disk** — диск;
- **worksp.** — рабочая область MatLab;

3) кнопка загрузки данных **Load Data...**, по нажатию которой появляется диалоговое окно выбора файла, если загрузка данных происходит с диска, или окно ввода идентификатора выборки, если загрузка данных происходит из рабочей области;

4) кнопка очистки данных **Clear Data**.

Примечание. В течение одного сеанса работы ANFIS-редактора можно загружать данные одного формата, т. е. количество входных переменных в выборках должно быть одинаковым.

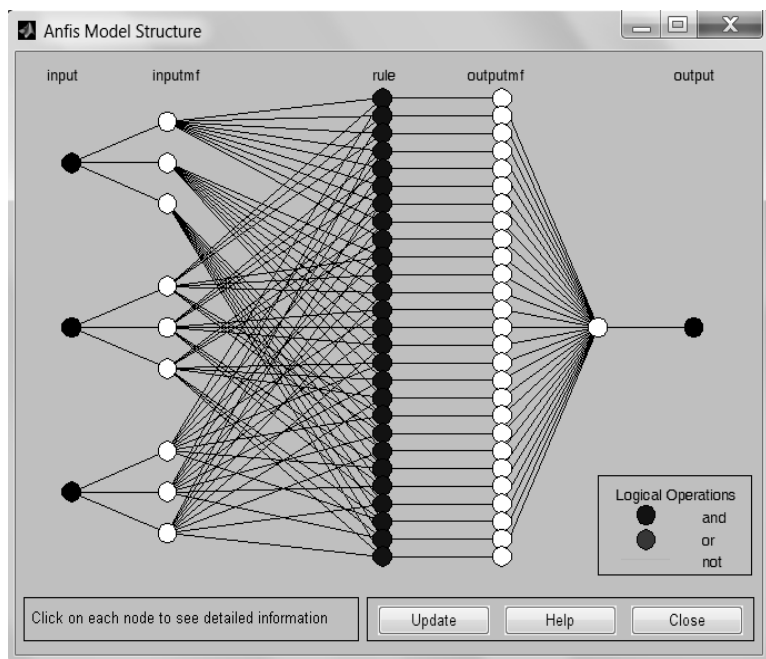


Рис. 5.6. Пример структуры нейро-нечеткой сети

Область генерирования исходной системы нечеткого логического вывода

В области генерирования (**Generate FIS**) (см. рис. 5.2) расположено меню выбора способа создания исходной системы нечеткого логического вывода. Меню содержит следующие альтернативы:

- **Load from disk** — загрузка системы с диска;
- **Load from worksp.** — загрузка системы из рабочей области MatLab;
- **Grid partition** — генерирование системы по методу решетки (без кластеризации);
- **Sub. clustering** — генерирование системы по методу субкластеризации.

В области также расположена кнопка **Generate**, по нажатию которой генерируется исходная система нечеткого логического вывода.

При выборе **Grid partition** появляется окно ввода параметров метода решетки (рис. 5.7), в котором нужно указать количество термов для каждой входной переменной и тип функций принадлежности для входных и выходной переменных.

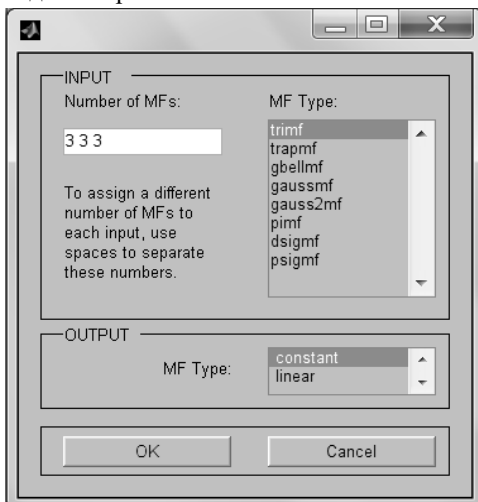


Рис. 5.7. Окно ввода параметров для метода решетки

При выборе **Load from disk** появляется стандартное диалоговое окно открытия файла.

При выборе **Load from worksp.** появляется стандартное диалоговое окно ввода идентификатора системы нечеткого логического вывода.

При выборе **Sub. clustering** появляется окно ввода следующих параметров метода субкластеризации (рис. 5.8):

- **Range of influence** — уровни влияния входных переменных;
- **Squash factor** — коэффициент подавления;
- **Accept ratio** — коэффициент, устанавливающий, во сколько раз потенциал данной точки должен быть выше потенциала центра первого кластера для того, чтобы центром одного из кластеров была назначена рассматриваемая точка;
- **Reject ratio** — коэффициент, устанавливающий, во сколько раз потенциал данной точки должен быть ниже потенциала центра первого кластера, чтобы рассматриваемая точка была исключена из возможных центров кластеров.

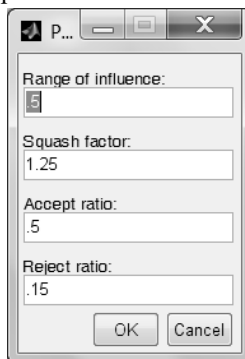


Рис. 5.8. Окно ввода параметров для метода субкластеризации

Область обучения

В области обучения (**Train FIS**) (см. рис. 5.2) расположены меню выбора метода оптимизации (**Optim. Method**), поле задания требуе-

мой точности обучения (**Error Tolerance**), поле задания количества итераций обучения (**Epochs**) и кнопка **Train Now**, нажатие которой запускает режим обучения. Промежуточные результаты обучения выводятся в область визуализации и в рабочую область MatLab. В ANFIS-редакторе реализованы два метода обучения:

1) **backpropagation** — метод обратного распространения ошибки, основанный на идеях метода наискорейшего спуска;

2) **hybrid** — гибридный метод, объединяющий метод обратного распространения ошибки с методом наименьших квадратов.

Область тестирования

В области тестирования (**Test FIS**) расположены меню выбора выборки и кнопка **Test Now**, по нажатии по которой происходит тестирование нечеткой системы с выводом результатов в область визуализации.

Область вывода текущей информации

В этой области выводится наиболее существенная текущая информация, например сообщения об окончании выполнения операций, значение ошибки обучения или тестирования и т. п.

5.4.2. Синтез нейро-нечеткой сети в среде MATLAB

Пример 1. Имеются исходные данные, представляющие собой входные переменные в виде массива целых случайных чисел в интервалах от 0 до 20 (первая переменная X_1), от 0 до 15 (вторая переменная X_2) и от 0 до 5 (третья переменная X_3). Выходная переменная вычисляется как $X_1 + X_2 - X_3$ (табл. 5.1). Требуется построить нейро-нечеткую сеть и проверить ее работу при различных значениях входных переменных.

Общая последовательность процесса разработки модели гибридной сети может быть представлена в следующем виде.

1. Подготовка файла с обучающими данными. Целесообразно воспользоваться редактором электронных таблиц MS EXCEL. Обучающую выборку необходимо сохранить во внешнем файле (файл сохраняется с расширением *.txt, которое затем меняется на *.dat).

2. Открыть редактор ANFIS. Загрузить файл с обучающими данными.

Таблица 5.1

Набор данных для обучения нейро-нечеткой сети

Первая входная переменная	Вторая входная переменная	Третья входная переменная	Выходная переменная
8	2	4	6
11	12	3	20
20	10	2	28
20	8	1	27
19	13	3	29
15	14	0	29
6	12	5	13
6	0	4	2
2	9	0	11
16	9	0	25
10	7	4	13
1	12	3	10
9	10	2	17
16	2	4	14
11	12	3	20
5	1	4	2
15	13	4	24
10	15	3	22
3	9	4	8
2	2	1	3
7	3	1	9
20	0	1	19
14	8	1	21
14	8	0	22
3	13	5	11
1	4	4	1
12	13	4	21
8	10	5	13
16	2	5	13
12	4	4	12

Кнопка загрузки данных **Load Data**, по нажатии которой появляется диалоговое окно выбора файла, если загрузка данных происхо-

дит с диска, или окно ввода идентификатора выборки, если загрузка данных происходит из рабочей области.

Внешний вид редактора ANFIS с загруженными обучающими данными изображен на рисунке 5.9.

3. После подготовки и загрузки обучающих данных можно сгенерировать структуру системы нечеткого вывода FIS типа Сугено, которая является моделью гибридной сети в системе MATLAB.

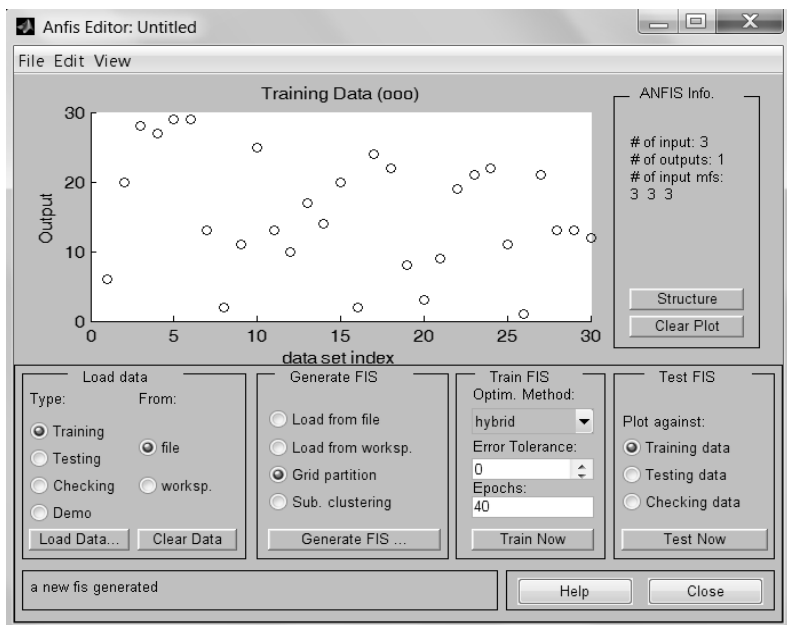


Рис. 5.9. Графический интерфейс редактора ANFIS после загрузки обучающих данных

Для этой цели следует воспользоваться кнопкой **Generate FIS** в нижней части рабочего окна редактора. При этом 2 первые опции относятся к предварительно созданной структуре гибридной сети, а 2 последних — к форме разбиения входных переменных модели.

Перед генерацией структуры системы нечеткого вывода типа Су-гено после вызова диалогового окна свойств зададим для каждой из входных переменных по 3 *лингвистических термина*, а в качестве типа их функций принадлежности выберем треугольные функции.

После нажатия кнопки **Generate FIS** вызывается диалоговое окно с указанием числа и типа функций принадлежности для отдельных термов входных переменных и выходной переменной (см. рис. 5.7).

4. После генерации структуры гибридной сети можно визуализировать ее структуру, для чего следует нажать кнопку **Structure** в правой части графического окна (см. рис. 5.6).

5. Перед обучением гибридной сети необходимо задать параметры обучения, для чего следует воспользоваться следующей группой опций в правой нижней части рабочего окна.

Выбрать метод обучения гибридной сети — обратного распространения (**backpropagation**) или гибридный (**hybrid**), представляющий собой комбинацию метода наименьших квадратов и метода убывания обратного градиента. Установить уровень ошибки обучения (**Error Tolerance**) — по умолчанию значение 0 (**изменять не рекомендуется**). Задать количество циклов обучения (**Epochs**) — по умолчанию значение 3 (рекомендуется увеличить для рассматриваемого примера, задать его значение равным 40). Для обучения сети следует нажать кнопку **Train Now**. При этом ход процесса обучения иллюстрируется в окне визуализации в форме графика зависимости ошибки от количества циклов обучения (рис. 5.10).

Дальнейшая настройка параметров построенной и обученной гибридной сети может быть выполнена с помощью стандартных графических средств пакета Fuzzy Logic Toolbox. Для этого необходимо воспользоваться командами **Membership Functions** (рис. 5.11) и **FIS Properties** (рис. 5.12) из меню **Edit**.

Для просмотра правил необходимо воспользоваться командой **Rules** из меню **View** (рис. 5.13) и **Edit** (рис. 5.14). Для тестирования и анализа полученной модели вводят различные входные данные. Первоначально проверку можно провести, пользуясь исходными данными.

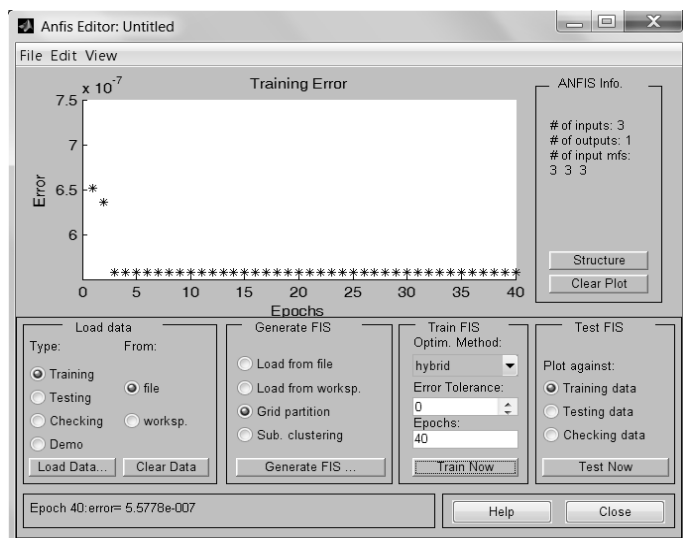


Рис. 5.10. График зависимости ошибок обучения от количества циклов обучения

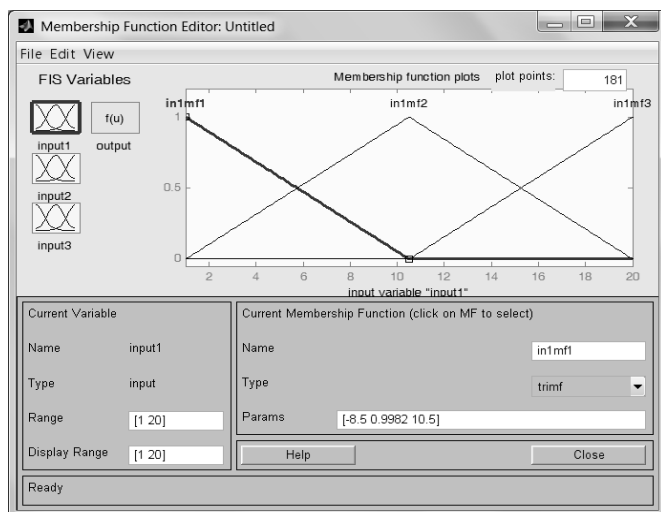


Рис. 5.11. Графический интерфейс редактора функций принадлежности построенной системы

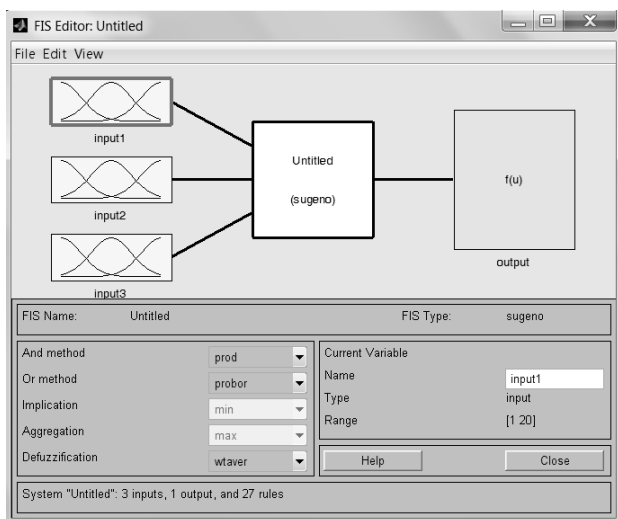


Рис. 5.12. Графический интерфейс редактора FIS для сгенерированной системы нечеткого вывода

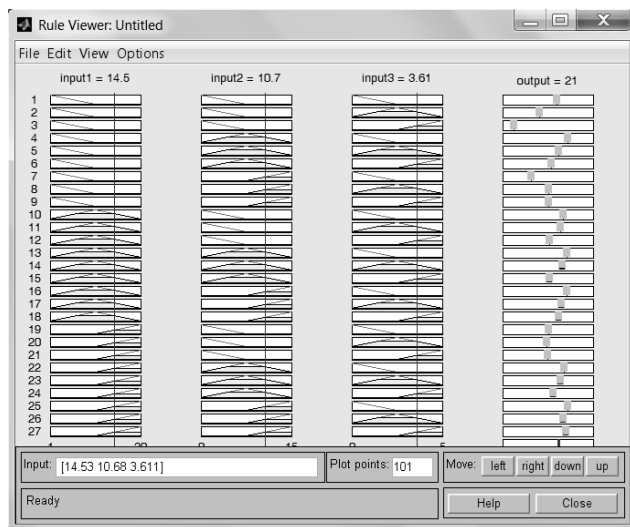


Рис. 5.13. Графический интерфейс просмотра правил сгенерированной системы

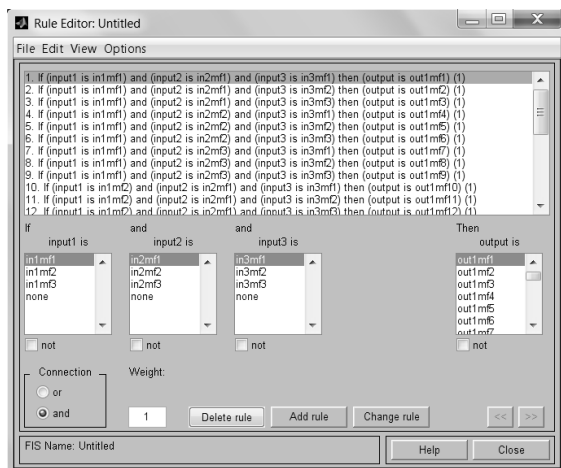


Рис. 5.14. Фрагмент базы нечетких правил

Для этого необходимо выбрать в окне редактора тип данных **Testing** и загрузить файл с данными нажатием кнопки загрузки данных **Load Data** (рис. 5.14). Для тестирования и анализа полученной модели можно воспользоваться также окном просмотра правил.

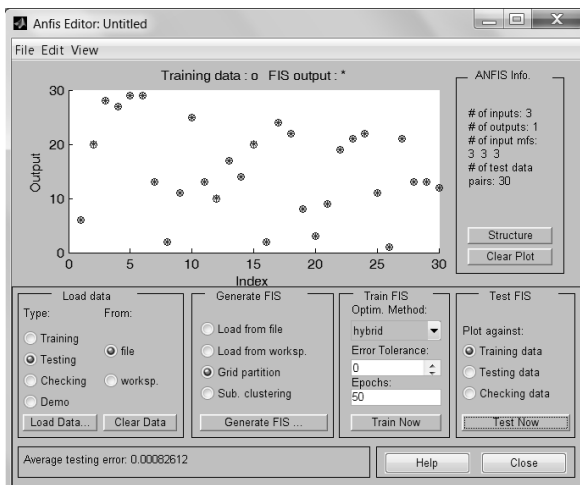


Рис. 5.15. Тестирование сгенерированной системы

Сначала введем числа, по которым обучалась система, в итоге получим верный результат. Затем введем числа, которых не было в данных для обучения, в итоге получим результат с погрешностью 0.06 (рис. 5.16). Те же самые операции можно проделать и в командном окне.

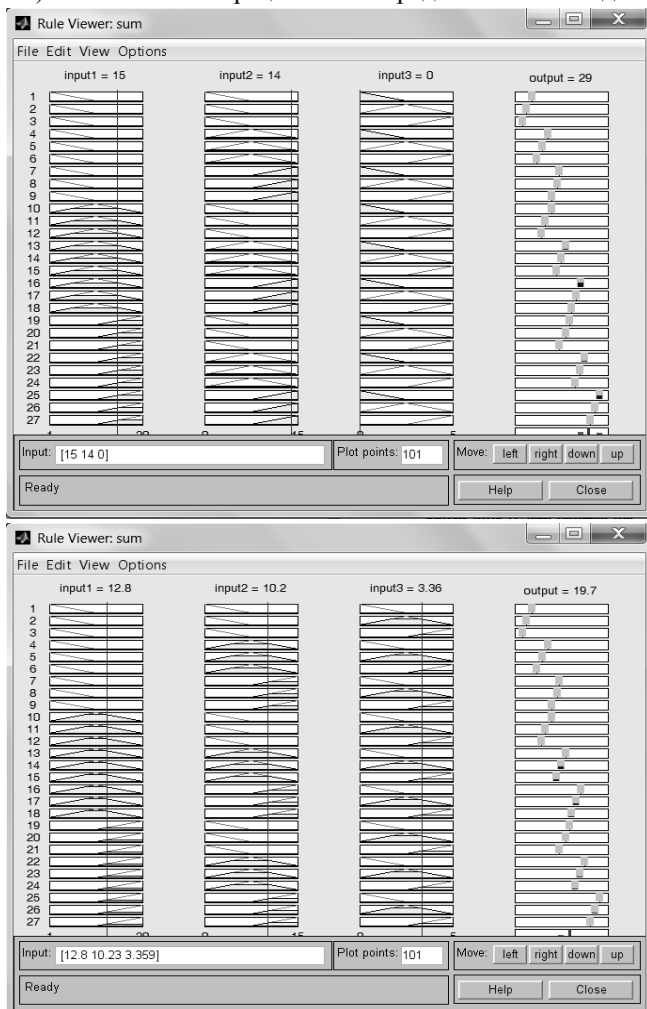


Рис. 5.16. Результаты тестирования разными числами

Для этого рекомендуется экспортировать созданную систему нечеткого вывода в рабочую область, задать вектор входных переменных и воспользоваться функцией `evalfis`:

```
>> x=[6,6,1];
y=evalfis(x,sum)
```

```
y =
```

```
11.0774
```

```
>> x=[5,4,2];
y=evalfis(x,sum)
```

```
y =
```

```
6.8910
```

§ 5.5. Задание для лабораторной работы

Задание 1.

1. Подготовить файл с обучающими данными с расширением ***.dat**, как указано в п. 5.4.2, по данным таблицы 5.1 с применением MS EXCEL.

2. Загрузить файл с обучающими данными в редактор ANFIS.

Задание 2.

1. Сгенерировать структуру системы нечеткого вывода FIS типа Сугено.

2. Произвести обучение нейро-нечеткой сети, предварительно задав параметры обучения.

Задание 3.

Провести проверку адекватности построенной нечеткой модели гибридной сети. Результаты внести в отчет в виде картинок изобра-

жений, полученных в MATLAB. Созданную сеть сохранить в MATLAB.

Задание 4.

1. Подготовить файл с обучающими данными с расширением *.dat для зависимости $y=x_1^2+x_2-x_3$ и загрузить его в редактор ANFIS.

2. Сгенерировать структуру системы нечеткого вывода FIS типа Сугено, произвести обучение нейро-нечеткой сети и провести проверку адекватности построенной нечеткой модели гибридной сети.

Указание. Для выполнения заданий следует использовать методику выполнения работ п. 5.4.2. Все решения по заданиям производить с использованием системы MATLAB. Для оформления работы использовать Microsoft WORD.

§ 5.6. Структура отчета

По материалам работы каждым студентом составляется отчет по установленной форме с использованием Microsoft WORD. Расчеты, диаграммы, графики следует выполнять с использованием ПЭВМ. Особое внимание при оформлении отчета студенты должны обратить на составление выводов по выполненной работе. В выводах нужно сопоставить результаты проведенных исследований с известными из теоретического курса закономерностями и выяснить согласованность полученных результатов с теоретическими. Полностью оформленный отчет представляется каждым студентом преподавателю на следующем занятии.

Отчет по лабораторной работе должен содержать:

- 1) титульный лист, оформленный по образцу;
- 2) цель лабораторной работы и исходные данные в Microsoft WORD и EXCEL;

3) результаты исследований по всем заданиям в виде таблиц, рисунков, графиков, структурных схем нейронных сетей; алгоритмов, текстов программ в Microsoft WORD и EXCEL;

4) выводы по заданиям.

К отчету прилагаются программы в системе MATLAB в электронном виде.

Представленные в отчете результаты, порядок их получения, расчетов, графиков и схем студенты обязаны уметь четко пояснить. За проведенную работу и оформленный отчет преподаватель выставляет дифференцированную оценку.

Приложение

СПИСОК ФУНКЦИЙ

NEURAL NETWORK TOOLBOX [13, 14]

1. Функции для анализа:

rrsurf — поверхность ошибки нейрона с единственным входом;
 maxlinlr — максимальная скорость обучения для линейного нейрона.

2. Функции отклонения:

boxdist — расстояние между двумя векторами;
 dist — евклидова весовая функция отклонения;
 linkdist — связанная функция отклонения;
 mandist — весовая функция отклонения Манхеттена.

3. Функции графического интерфейса:

nntool — вызов графического интерфейса пользователя.

4. Функции инициализации слоя:

initnw — функция инициализации Нгуен — Видроу (Nguyen — Widrow);
 initwb — функция инициализации по весам и смещениям.

5. Функции обучения:

learncon — обучающая функция смещений;
 learngd — обучающая функция градиентного спуска;
 learnngdm — обучающая функция градиентного спуска с учетом моментов;
 learnh — обучающая функция Хэбба;
 learnhd — обучающая функция Хэбба с учетом затухания;
 learnis — обучающая функция instar;
 learnk — обучающая функция Кохонена;
 learnlv1 — обучающая функция LVQ1;

learnlv2 — обучающая функция LVQ2;

learnos — обучающая функция outstar;

learnp — обучающая функция смещений и весов перцептрона;

learnpn — обучающая функция нормализованных смещений и весов перцептрона;

learnsom — обучающая функция самоорганизующейся карты весов;

learnwh — правило обучения Уидроу — Хоффа (Widrow — Hoff).

6. Линейные функции поиска:

srchbac — одномерная минимизация с использованием поиска с возвратом;

srchbre — одномерная локализация интервала с использованием метода Брента (Brent);

srchcha — одномерная минимизация с использованием метода Караламбуца (Charalambous);

srchgol — одномерная минимизация с использованием золотого сечения;

srchhyb — одномерная минимизация с использованием гибридного бисекционного поиска.

7. Функции вычисления производных от входов сети:

dnetprod — вычисление производной от входов сети с перемножением входов;

dnetsum — вычисление производной от входов сети с суммированием входов.

8. Входные функции сети:

netprod — функция произведения входов;

netsum — функция суммирования входов.

9. Функции инициализации сети:

initlay — функция послойной инициализации сети.

10. Функции использования сети:

adapt — разрешает адаптацию сети;
 disp — отображает свойства нейронной сети;
 display — отображает имена переменных и свойства сети;
 init — инициализация нейронной сети;
 sim — моделирование нейронной сети;
 train — тренировка нейронной сети.

11. Функции создания новой сети:

network — создание нейронной сети пользователя;
 newsc — создание конкурентного слоя;
 newcfc — создание каскадной направленной сети;
 newelm — создание сети обратного распространения Элмана (Elman);
 newff — создание однонаправленной сети;
 newfftd — создание однонаправленной сети с входными задержками;
 newgrnn — создание обобщенной регрессионной нейронной сети;
 newhop — создание рекуррентной сети Хопфилда;
 newlin — создание линейного слоя;
 newlind — конструирование линейного слоя;
 newlvq — создание квантованной сети;
 newp — создание персептрона;
 newpnn — конструирование вероятностной нейронной сети;
 newrb — конструирование сети с радиальным базисом;
 newrbe — конструирование точной сети с радиальными базисными функциями;
 newsom — создание самоорганизующейся карты.

12. Функции производных функционирования:

dmae — средняя абсолютная ошибка вычисления производной;
 dmse — среднеквадратичная ошибка производной;
 dmsereg — среднеквадратичная ошибка производной w/reg;
 dsse — суммарная квадратичная ошибка производной.

13. Функции выполнения:

mae — средняя абсолютная ошибка;
 mse — среднеквадратичная ошибка;
 msereg — среднеквадратичная ошибка w/reg;
 sse — суммарная квадратичная ошибка.

14. Функции графики:

hintonw — график Хинтона для матрицы весов;
 hintonwb — график Хинтона для матрицы весов и векторов смещений;
 plotbr — график функционирования сети при регулярной тренировке (Bayesian);
 ploter — изображение положений весов и смещений на поверхности ошибки;
 plotes — изображение поверхности ошибок единичного входного нейрона;
 plotpc — изображение линии классификации в векторном пространстве перцептрона;
 plotperf — графическое представление функционирования сети;
 plotpv — графическое представление входных целевых векторов;
 plotsom — графическое представление самоорганизующейся карты;
 plotv — графическое представление векторов в виде линий, выходящих из начала координат;
 plotvec — графическое представление векторов различными цветами.

15. Функции предварительной и постобработки:

postmnmx — ненормализованные данные, которые были нормализованы посредством prenmnx;
 postreg — линейный регрессионный анализ выходов сети по отношению к целевым значениям обучающего массива;
 poststd — ненормированные данные, которые были нормированы с помощью функции prestd;

premnmx — нормирование данных в диапазоне от -1 до $+1$;

prepsa — анализ главных компонент для входных данных;

prestd — нормирование данных к единичному стандартному отклонению и нулевому среднему;

tramnmx — преобразование данных с предварительно вычисленными минимумом и максимумом;

trapca — преобразование данных с использованием PCA матрицы, вычисленной с помощью функции prepsa;

trastd — преобразование данных с использованием предварительно вычисленных значений стандартного отклонения и среднего.

16. Функции поддержки Simulink:

gensim — генерация блока Simulink для моделирования нейронной сети.

17. Топологические функции:

gridtop — топологическая функция в виде сеточного слоя;

hextop — топологическая функция в виде гексагонального слоя;

randtop — топологическая функция в виде случайного слоя.

18. Функции тренировки:

trainb — пакетная тренировка с использованием правил обучения для весов и смещений;

trainbfg — тренировка сети с использованием квазиньютоновского метода BFGS;

trainbr — регуляризация Bayesian;

trainc — использование приращений циклического порядка;

traincgb — метод связанных градиентов Пауэлла — Била (Powell — Beale);

traincgf — метод связанных градиентов Флетчера — Пауэлла (Fletcher — Powell);

traincgp — метод связанных градиентов Полака — Рибера (Polak — Ribiere);

traingd — метод градиентного спуска;
 traingda — метод градиентного спуска с адаптивным обучением;
 traingdm — метод градиентного спуска с учетом моментов;
 traingdx — метод градиентного спуска с учетом моментов и с адаптивным обучением;
 trainlm — метод Левенберга — Маркара (Levenberg — Marquardt);
 trainoss — одноступенчатый метод секущих;
 trainr — метод случайных приращений;
 trainrp — алгоритм упругого обратного распространения;
 trains — метод последовательных приращений;
 trainscg — метод шкалированных связанных градиентов.

19. Производные функций активации:

dhardlim — производная ступенчатой функции активации;
 dhardlms — производная симметричной ступенчатой функции активации;
 dlogsig — производная сигмоидной (логистической) функции активации;
 dposlin — производная положительной линейной функции активации;
 dpurelin — производная линейной функции активации;
 dradbas — производная радиальной базисной функции активации;
 dsatlin — производная насыщающейся линейной функции активации;
 dsatlins — производная симметричной насыщающейся функции активации;
 dtansig — производная функции активации гиперболический тангенс;
 dtribas — производная треугольной функции активации.

20. Функции активации:

compet — конкурирующая функция активации;
 hardlim — ступенчатая функция активации;

hardlims — ступенчатая симметричная функция активации;

logsig — сигмоидная (логистическая) функция активации;

poslin — положительная линейная функция активации;

purelin — линейная функция активации;

radbas — радиальная базисная функция активации;

satlin — насыщающаяся линейная функция активации;

satlins — симметричная насыщающаяся линейная функция активации;

softmax — функция активации, уменьшающая диапазон входных значений;

tansig — функция активации гиперболический тангенс;

tribas — треугольная функция активации.

21. Полезные функции:

calca — вычисляет выходы сети и другие сигналы;

calcal — вычисляет сигналы сети для одного шага по времени;

calce — вычисляет ошибки слоев;

calcel — вычисляет ошибки слоев для одного шага по времени;

calcgx — вычисляет градиент весов и смещений как единственный вектор;

calcjejj — вычисляет Якобиан;

calcjx — вычисляет Якобиан весов и смещений как одну матрицу;

calcpd — вычисляет задержанные входы сети;

calcperf — вычисление выходов сети, сигналов и функционирования;

formx — формирует один вектор из весов и смещений;

getx — возвращает все веса и смещения сети как один вектор;

setx — устанавливает все веса и смещения сети в виде одного вектора.

22. Векторные функции:

cell2mat — объединяет массив элементов матриц в одну матрицу;

combvec — создает все комбинации векторов;

con2seq — преобразует сходящиеся векторы в последовательные векторы;

concur — создает сходящиеся векторы смещений;

ind2vec — преобразование индексов в векторы;

mat2cell — разбиение матрицы на массив элементов матриц;

minmax — вычисляет минимальные и максимальные значения строк матрицы;

normc — нормирует столбцы матрицы;

normr — нормирует строки матрицы;

pnormc — псевдонормировка столбцов матрицы;

quant — дискретизация величины;

seq2con — преобразование последовательных векторов в сходящиеся векторы;

sumsq — сумма квадратов элементов матрицы;

vec2ind — преобразование векторов в индексы.

23. Функции инициализации весов и смещений:

initcon — «сознательная» функция инициализации;

initzero — инициализация с установкой нулевых значений весов и смещений;

midpoint — инициализация с установкой средних значений весов;

randnc — инициализация с установкой нормализованных значений столбцов весовых матриц;

randnr — инициализация с установкой нормализованных значений строк весовых функций;

randn — инициализация с установкой симметричных случайных значений весов и смещений;

revert — возвращение весам и смещениям значений, соответствующих предыдущей инициализации.

24. Функции весовых производных:

ddotprod — производная скалярного произведения.

25. Весовые функции:

dist — евклидово расстояние;

dotprod — весовая функция в виде скалярного произведения;

mandist — весовая функция — расстояние Манхеттена;

negdist — весовая функция — отрицательное расстояние;

normprod — нормированное скалярное произведение.

СПИСОК ЛИТЕРАТУРЫ

1. *Романов, П. С.* Искусственный интеллект: аспекты проблемы создания // Вестник Коломенского института (филиала) Московского политехнического университета. Серия «Естественные и технические науки». — 2018. — № 11. — С. 104–115.

2. *Ясницкий, Л. Н.* Искусственный интеллект : учебное пособие. — М. : Бином, 2011. — 200 с.

3. *Джонс, Т. М.* Программирование искусственного интеллекта в приложениях : пер. с англ. А. И. Осипова. — М. : ДМК Пресс, 2011. — 312 с.

4. *Романов, П. С.* Системы искусственного интеллекта : учебное пособие / П. С. Романов, И. П. Романова ; под общ. ред. П. С. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — 244 с.

5. *Романов, П. С.* Искусственный интеллект и нанотехнологии: взаимосвязь и взаимовлияние / П. С. Романов, И. П. Романова // Энергия: экономика, техника, экология. — 2020. — № 9. — С. 24–31.

6. *Романов, П. С.* Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 2 : Моделирование персептрона и линейных нейронных сетей в системе MATLAB. — 76 с.

7. *Романов, П. С.* Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 3 : Моделирование нейронных сетей в системе MATLAB. — 115 с.

8. *Романова, И. П.* Системы искусственного интеллекта (практикум) : учебное пособие / И. П. Романова, П. С. Романов ; под общ. ред. П. С. Романова. — Коломна : КИ(ф) МГОУ, 2014. — 86 с.

9. *Романова, И. П.* Системы искусственного интеллекта : учебное пособие (контрольная работа) / И. П. Романова, П. С. Романов ; под общ. ред. П. С. Романова. — Коломна : КИ(ф) МГОУ, 2015. — 38 с.

10. *Романов, П. С.* Системы искусственного интеллекта (лабораторный практикум) : учебное пособие / П. С. Романов, И. П. Романова.

ва. — Коломна : ГСГУ ; КИ(ф) МПУ, 2017. — Часть 1 : Моделирование нечетких систем в системе MATLAB. — 115 с.

11. *Тимохин, А. Н.* Моделирование систем управления с применением MatLab : учебное пособие / А. Н. Тимохин, Ю. Д. Румянцев. — М. : НИЦ ИНФРА-М, 2016. — 256 с.

12. *Дьяконов, В. П.* MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики / В. П. Дьяконов, В. В. Круглов. — М. : СОЛОН-ПРЕСС, 2006. — 456 с.

13. *Дьяконов, В. П.* MATLAB 7.*/R2006/R2007: самоучитель. — М. : ДМК Пресс, 2008. — 768 с.

14. *Леоненков, А. В.* Нечеткое моделирование в среде MATLAB и fuzzyTECH. — СПб. : БХВ-Петербург, 2005. — 736 с.

15. *Поршнев, С. В.* MATLAB 7. Основы работы и программирования : учебник. — М. : Бином-Пресс, 2011. — 320 с.

16. *Усков, А. А.* Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика / А. А. Усков, А. В. Кузьмин. — М. : Горячая линия — Телеком, 2004. — 143 с.

17. *Штовба, С. Д.* Проектирование нечетких систем средствами MATLAB. — М. : Горячая линия — Телеком, 2007. — 288 с.

18. Моделирование нейронных сетей в MATLAB : методические указания к лабораторной работе / сост. А. В. Федотов. — Омск : Изд-во ОГТУ, 2010. — 14 с.

19. Моделирование искусственных нейронных сетей в системе MATLAB / сост. Д. А. Донской, Б. Д. Шашков, Д. М. Деревянчук [и др.]. — Пенза : Изд-во: ПГУ, 2005. — Часть 2 : Линейные сети. Методические указания к лабораторным работам. — 33 с.

20. Нейро-нечеткое моделирование в среде MATLAB : методические указания к лабораторным работам / сост. Л. Р. Черняховская, Р. А. Шкундина, И. В. Осипова, И. Б. Герасимова. — Уфа : Уфимск. гос. авиац. техн. ун-т., 2004. — 20 с.