МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по выполнению лабораторных работ
по дисциплине
«ТЕОРИЯ НЕЙРОННЫХ СЕТЕЙ»
для направления подготовки **09.04.02 Информационные системы и технологии**
направленность (профиль) **Технологии работы с данными и знаниями, анализ информации**

Пятигорск
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ....3
1. Цель и задачи изучения дисциплины....3
2. Оборудование и материалы....3
3. Наименование лабораторных работ....3
4. Содержание лабораторных работ....4
Лабораторная работа № 1....4
Лабораторная работа № 2....9
Лабораторная работа № 3....17
Лабораторная работа № 4....21
Лабораторная работа № 5....26
КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ....30
МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ....31
УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 31

ВВЕДЕНИЕ

1. Цель и задачи изучения дисциплины

Целью освоения дисциплины «Теория нейронных сетей» является овладение необходимыми теоретическими знаниями и практическими навыками в области современных методов и подходов параллельной обработки информации для задач построения современных информационных сетей и систем.

Задачами курса являются: знакомство с моделями операционного блока нейрокомпьютера; изучение основных моделей нейронных сетей; изучение основ конструирования и настроек нейронных сетей; применение методов обучения нейронных сетей; изучение программной эмуляции нейрокомпьютеров и нейронных сетей.

2. Оборудование и материалы

Для проведения практических занятий необходимо следующее материально-техническое обеспечение: персональный компьютер; проектор; возможность выхода в сеть Интернет для поиска по образовательным сайтам и порталам; интерактивная доска.

3. Наименование лабораторных работ

Лабораторная работа 1.
Тема 1: Биологические основы нейросетей. Классификация нейронных сетей.
Содержание: Знакомство с системой MatLab.
Лабораторная работа 2.
Тема 2: Методы обучения нейронных сетей. Архитектура нейронных сетей.
Содержание: Знакомство с инструментальным средством Neural Network Toolbox.
Создание и обучение нейронной сети для реализации функции двух переменных.
Лабораторная работа 3.
Тема 4: Программное средство Neural Network Toolbox системы MatLab.
Содержание: Изучение свойств персептронов. Создание и обучение нейронной сети для реализации логической функции.
Лабораторная работа 4.
Тема 5: Персептрон. Линейные нейронные сети. Радиальные базисные сети.
Содержание: Исследование линейных сетей прямого распространения сигнала
Лабораторная работа 5.
Тема 6: Самоорганизующиеся и рекуррентные сети. Гибридные нейронные сети.
Содержание: Исследование самоорганизующихся нейронных сетей Кохонена.

4. Содержание лабораторных работ

Лабораторная работа № 1.

Биологические основы нейросетей. Классификация нейронных сетей.
Форма проведения: компьютерное моделирование
Цель: Знакомство с системой MatLab.

Теоретические обоснования

При работе в **Windows** запустить систему MATLAB можно из меню **Пуск** (стартового меню) этой операционной системы. Альтернативным вариантом запуска MATLAB является двойной щелчок на ярлыке системы MATLAB, расположенном на рабочем столе **Windows**. Запуск MATLAB 6.x отображает на экране окно рабочей среды, подобное показанному на рис. 1.

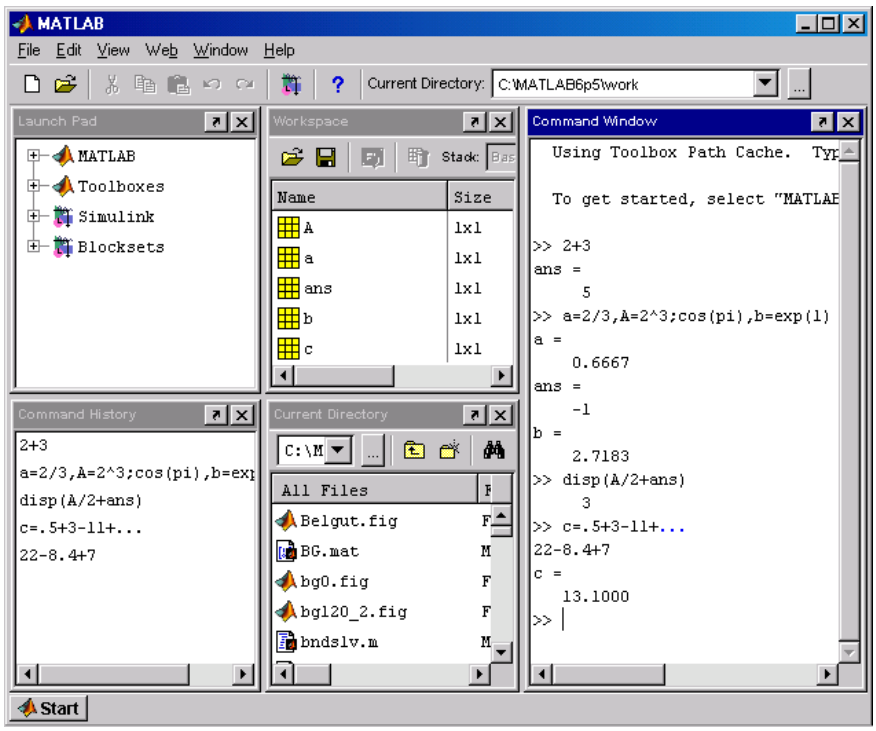


Рис 1.

Окно рабочей среды (графический интерфейс) состоит из следующих основных элементов:
строка меню;
панель инструментов с кнопками и раскрывающимся списком;
окно с **Launch Pad** (Панель запуска) содержит дерево файловой системы, где отображены только установленные на компьютере разделы расширений системы MATLAB. С помощью этого окна можно запустить любой из них;
окно **Workspace** (Рабочее пространство), из которого можно получить простой доступ к переменным, используемым в данном сеансе работы;
окно **Command History** (История команд), предназначенное для просмотра и повторного вызова ранее введенных команд;
окно **Current Directory** (Текущий каталог), в котором отображается список файлов и вложенных папок активного в данный момент каталога;
окно **Command Window** (Окно команд) предназначено для ввода чисел, переменных, выражений и команд, для промотра результатов вычислений, для отображения текстов выполняемых программ, а также для вывода сообщений об ошибках;
строка состояния, где отображаются сообщения системы.
Пользователь может настроить окно рабочей среды по своему усмотрению. Можно, например, изменить местоположение и размер внутренних окон приемами, общими для **Windows** - приложений. Отобразить или скрыть соответствующие окна можно с помощью команд меню **View** (Вид) основного меню MATLAB. Например, для отображения полной рабочей среды (рис. 1.1) надо выбрать команду **View => Desktop Layout => Five Panel**. Любое из внутренних окон полной рабочей среды можно закрыть щелчком по кнопке с крестиком в правом верхнем углу.

Арифметические вычисления

Работа в среде MATLAB может осуществляться либо в *программном* режиме, либо в *командном* режиме (режиме *калькулятора*, *диалоговом* режиме) по правилу «задал вопрос, получил ответ». Это превращает MATLAB в необычайно мощный калькулятор, который способен производить не только обычные для калькулятора вычисления, но и

операции с векторами и матрицами, комплексными числами, рядами и полиномами. Можно почти мгновенно задать и вывести графики различных функций: от простой синусоиды до сложной трехмерной фигуры.

Основным элементом командного режима работы с системой является главное или командное окно **Command Window**. Оно активизируется командой **View => Desktop Layout => Command Window Only** основного меню MATLAB. Структура командного окна аналогична структуре **Windows** - приложений (рис. 2).

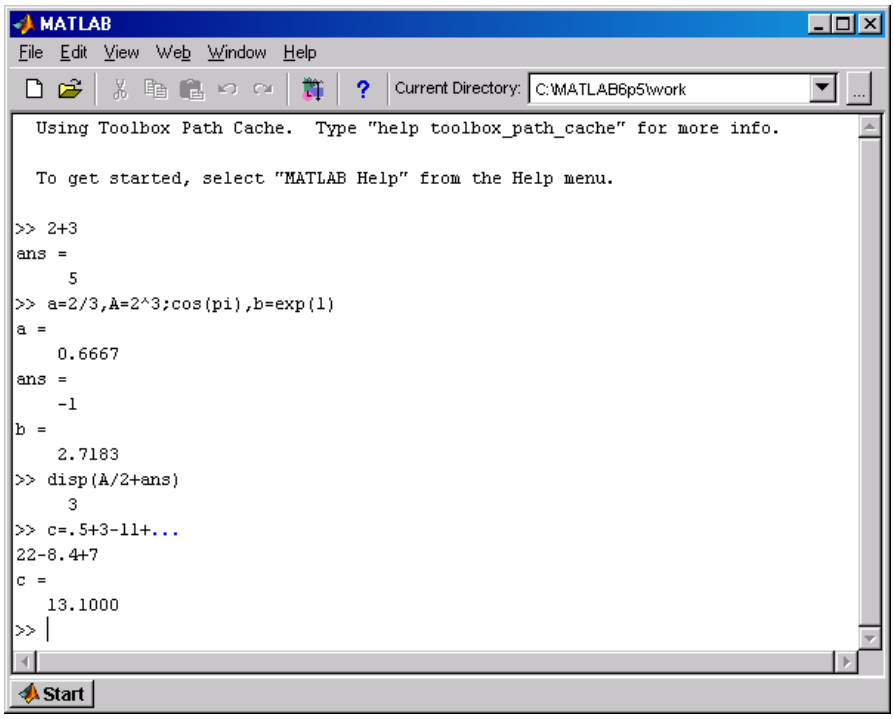


Рис. 2

Строка в текстовом поле командного окна, отмеченная символом приглашения >>с мигающим курсором, называется *строкой ввода* или *командной строкой*. Она предназначена для ввода с клавиатуры команд, чисел, имен переменных и знаков операций, составляющих выражение. Для того, чтобы система MATLAB выполнила введенную команду или вычислила заданное выражение, следует нажать клавишу **<Enter>**(Ввод).

При вводе курсор может находиться в любом месте командной строки. Введенные выражения вычисляются, а результаты вычислений и выполнения команд появляются в одной или нескольких строках командного окна – строках вывода.

В результате многократных вычислений (нажатий клавиши **<Enter>**) в командном окне автоматически производится *вертикальная протяжка (scrolling)*: строки сдвигаются на одну позицию вверх, а внизу появляется строка ввода с символом приглашения >>. Информация, которая покинула видимую часть окна, не исчезает. В MATLAB ранее введенные строки команд представляют собой «историю» и запоминаются в *стеке команд*.

Для просмотра выполненных команд и результатов вычислений, не умещающихся на экране, имеются полосы горизонтальной и вертикальной протяжки. Использование полос протяжки ничем не отличается от других **Windows** - приложений. Можно также осуществлять протяжку командного окна с помощью клавиш **<PageUp>, < PageDown>, <Ctrl+Home>**и **<Ctrl+End>.**

Клавиши <↑> и <↓>, которые в текстовых редакторах служат для перемещения вверх или вниз по экрану, в MATLAB работают иначе. Они используются для возврата в строку ввода ранее выполненных команд с целью их повторного выполнения или

редактирования. После первого нажатия клавиши <↑> в строке ввода отобразится последняя введенная команда, при втором нажатии – предпоследняя и т. д. Клавиша <↓> осуществляет прокрутку команд в противоположном направлении.

Иными словами, текстовое поле окна **Command Window** располагается в двух принципиально разных зонах: *зоне просмотра* и *зоне редактирования*. Зона редактирования находится в командной строке, а вся остальная информация видимой части командного окна – в зоне просмотра.

Пока не нажата клавиша **<Enter>**, вводимое выражение может быть отредактировано или удалено. В зоне просмотра уже ничего нельзя исправить. Если поместить в нее курсор и нажать какую - либо клавишу на клавиатуре, курсор будет автоматически перемещен в строку ввода, расположенную в зоне редактирования. В то же время, с помощью клавиш <←> и <→> можно перемещать курсор в командной строке.

Невозможность редактирования ранее введенной команды простой установкой курсора в нужную строку является одной из особенностей системы MATLAB.

Сеанс работы с системой MATLAB называется *сессией*. Иными словами, сессия – это все то, что отображается в командном окне в процессе работы с системой. Команды сессии автоматически образуют список, который выводится в окне **Command History**, а значения переменных сохраняются в окне **Workspase** (рис. 1).

Например, сессия на рис. 2 отображает результаты последовательного ввода четырех команд. Обсудим эти результаты и отметим некоторые *особенности вычислений в системе* MATLAB:

```
>>2+3
ans =
5
```

Результату выполненной операции не было присвоено имя, поэтому при выводе он был автоматически обозначен символом **ans** (ansver – ответ). Под этим именем результат вычислений хранится в памяти компьютера и его можно использовать в последующих вычислениях до тех пор, пока в ходе работы не будет получен новый непоименованный результат. Результат вычислений выводится в строках вывода, не содержащих знака приглашения **>>**;

```
>>a=2/3,A=2^3;cos(pi),b=exp(1)
a =
0.6667
ans =
-1
b =
2.7183
```

В одной командной строке можно ввести несколько команд, разделяя их запятыми либо точками с запятой. Система MATLAB выполняет каждую команду, за которой следует запятая, и отображает результаты в отдельных строках. Результат выполнения команды, за которой следует символ <**;**>, на экран не выводится, но он сохраняется в памяти и может быть использован в последующих вычислениях.

Знаком присваивания является знак =, а не комбинированный знак **:=**, принятый, например, в языке программирования **Pascal** или в системе символьной математики **Maple**.

После ввода этой командной строки вычисляются и сохраняются в памяти значения выражений $a=2/3=0{,}6667$, $A=2^3=8$, $ans=\cos\pi=-1$, $b=e^1=e=2{,}7183$ (e – основание натурального логарифма). Значение переменной *A*, в отличие от *a*, *ans*, *b*, не выводится на экран из-за символа <**;**>. При вычислении $\cos\pi$ использовалась системная переменная *pi* – число π. Число *e* системной переменной не является, и для его вычисления использована

встроенная элементарная функция **exp(1)**. Функции записываются строчными буквами, а их аргументы указываются в круглых скобках. Аргумент встроенной тригонометрической функции **cos** задан в радианах;

>>disp(A/2+ans)

3

Команда **disp** (от слова «дисплей») вычисляет выражение$2^3/2+cos\pi$и выводит ответ, но не присваивает его переменной**ans**, как при обычных вычислениях:

>> A/2+ans

ans=

3

В дальнейшем **disp** используется для предотвращения вывода лишней строки **ans** =в наглядных документах;

>> c=.5+3-11+...

22-8.4+7

c =

13.1000

Иногда требуется ввести в окне **Command Window** команду, которая слишком длинна, чтобы уместиться на одной строке. При приближении к концу строки можно ввести**…**(три последовательные точки), нажать клавишу**<Enter>** и продолжить набор команды на следующей строке. При этом вы не увидите на новой строке символа приглашения**>>**.

Сессия на рис.1.2 содержит только правильные команды и результаты их выполнения. В общем случае сессия является результатом проб и ошибок. Ее текст, наряду с правильными определениями, содержит сообщения и предупреждения об ошибках, переопределения функций и переменных, использованную справочную информацию команды **help**. Если сессия сильно «засорена» лишней информацией, диалог пользователя с системой затрудняется.

Команда очистки экрана **clc**

>>clc

стирает содержание командного окна MATLAB и размещает символ приглашения >> в левом верхнем углу пустого экрана.

Эта команда, однако, оставляет неизменным содержимое окон **Command History** и **Workspase**. Поэтому в «чистом» командном окне можно пользоваться значениями переменных, полученных до ввода команды **clc**.

Если же появится необходимость отредактировать или повторить ранее выполненную команду, то это легко осуществить с помощью окна **Command History**.

Переменные – это именованные объекты, хранящие какие – либо данные.

Переменные могут быть числовыми, матричными или символьными, что зависит от типа хранящихся в них данных. Типы переменных заранее не декларируются. Они определяются выражением, значение которого присваиваивается переменной, т.е. пользователь не должен заботиться о том, какие значения будет принимать переменная (комплексные, вещественные или целые).

Имя переменной (ее *идентификатор*) может содержать до *31* символа и не должно совпадать с именами других переменных, функций, команд и системных переменных MATLAB. Имя переменной должно начинаться с буквы, может содержать цифры и символ подчеркивания. Среда MATLAB чувствителена к регистру букв (переменные *a* и *A* не идентичны).

В MATLAB существует несколько имен переменных, являющихся зарезервированными. Переменные с такими именами называются *системными*. Они задаются после загрузки системы и могут использоваться в арифметических выражениях.

Системные переменные могут быть переопределены, т. е. при небходимости им можно присвоить другие значения.
Ниже перечислены основные системные переменные MATLAB:
ans – результат вычисления последнего не сохраненного пользователем выражения;
i, **j** – мнимая единица (), используемая для задания мнимой части комплексных чисел;
Inf (infinity) – обозначение машинной бесконечности;
NaN – сокращение от слов Not-a-Number (не число), принятое для обозначения неопределенного результата (например, 0/0 или Inf/Inf).
pi – число π *(π=3,141592653589793)*;
eps – погрешность операций над числами с плавающей точкой, т.е интервал между числом *1.0* и следующим ближайшим числом с плавающей точкой равен *2.2204e-16* или 2^{-52};
realmin – минимальное по модулю вещественное число (*2.2251e-308* или 2^{-1022});
realmax – наибольшее по модулю вещественное число (*1.7977e+308* или 2^{1023}).
Приоритеты *арифметических операций*системы MATLAB в порядке убывания следующие:
1. Возведение в степень <^>.
2. Умножение <*> и деление (слева направо </>, справа налево <\>).
3. Сложение <+> и вычитание <–>.
Выполнение операций одинакового приоритета происходит в порядке слева направо. Для изменения порядка выполнения арифметических операторов следует использовать круглые скобки. Кроме арифметических операторов, в MATLAB имеются операторы отношения и логические операторы (см. разд. 4.1).
Полный список операторов и справочную информацию по любому из них можно получить в разделе **ops** справочной системы MATLAB, используя команды **help** или **doc** (см. Приложение 1).
Основу большинства расчетов составляют вычисления значений *арифметических выражений*. В них в качестве операндов могут выступать константы, переменные или функции. В отличие от большинства алгоритмических языков, в MATLAB допускается использование операндов - массивов (см. разд. 1.6, 1.7, 1.10). В этом случае результатом вычисления выражения также может быть массив.
Выражения, помещенные между двумя апострофами (заключенные в символьные скобки **'** **'**), рассматриваются как строчные и не вычисляются, даже если в них содержатся математические выражения. Чаще всего они применяются для задания параметров функций и их нечисловых значений, вставки текста в графические объекты, а также для описания символьных переменных и выражений. Так, ввод строки *'2+3'*приводит к результату

```
>>'2+3'
ans =
2+3
```

а не *5*.
При выводе графиков символы, помещенные между апострофами, определяют цвет линий графика, их тип и тип маркера, которым метятся линии

Лабораторная работа № 2.

Методы обучения нейронных сетей. Архитектура нейронных сетей.
Форма проведения: компьютерное моделирование.
Цель: Знакомство с инструментальным средством Neural Network Toolbox. Создание и обучение нейронной сети для реализации функции двух переменных.

Порядок выполнения работы

1. Ознакомиться с демонстрационными моделями:

nnd2n1 – одновходовой нейрон;
nnd2n2 – двухвходовой нейрон.

2. Ознакомиться с возможностями инструментального средства NNTool.

3. Выполнить демонстрационный пример из методических указаний и подобный пример согласно варианту.

4. Ознакомиться с моделью нейронной сети в среде Simulink.

5. Ответить на контрольные вопросы и составить отчет.

Теоретические обоснования

NNTool – инструментальное средство с графическим интерфейсом, позволяющее начинающему пользователю пакета Neural Network Toolbox выполнять создание, обучение, моделирование, экспорт и импорт нейронных сетей и данных, не обращаясь к командному окну MATLAB. Вызов NNTool возможен либо с помощью команды nntool из командной строки MATLAB, либо из окна запуска приложений Launch Pad с помощью опции NNTool из раздела Neural Network Toolbox. После вызова на экране дисплея появляется окно Network/Data Manager (Управление сетью/Данными) (рис. 1).
Элементы окна:
Help – кнопка вызова окна подсказки Network/ Data Manager Help;
New Data – кнопка вызова окна формирования данных Create New Data (рис. 2);

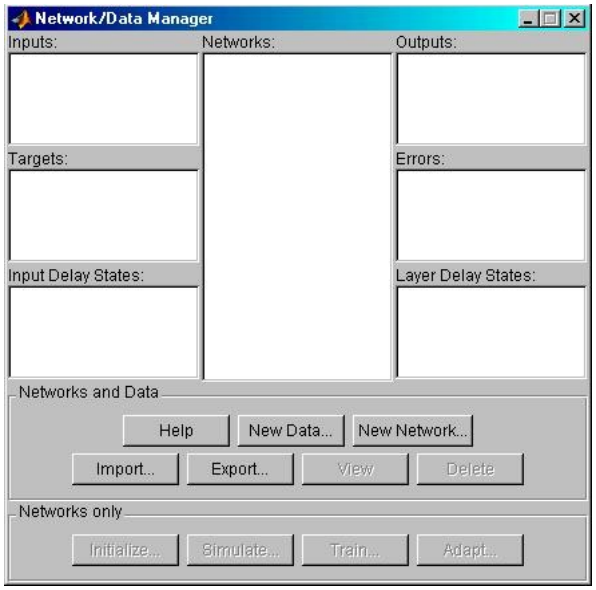


Рис. 1

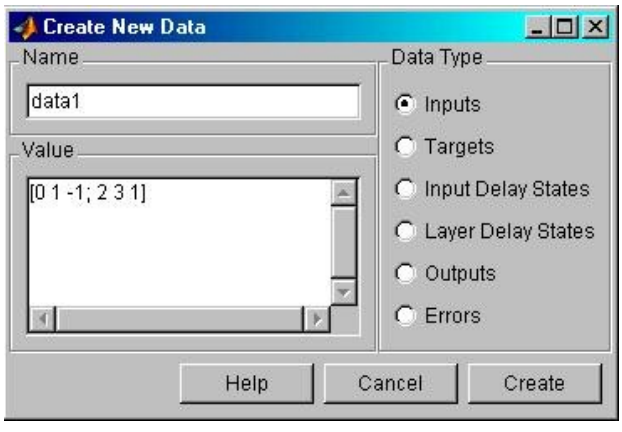


Рис. 2

New Network…– кнопка вызова окна создания нейронной сети Create New Network (рис. 3);

Import… – кнопка вызова окна для импорта или загрузки данных Import or Load to Network/Data Manager (рис. 4);

Export… – кнопка вызова окна для экспорта или записи данных в файл Export or Save from Network/Data Manager;

Кнопки View, Delete становятся активными после создания и активизации данных, относящихся к входам, целям, выходам или ошибкам сети.

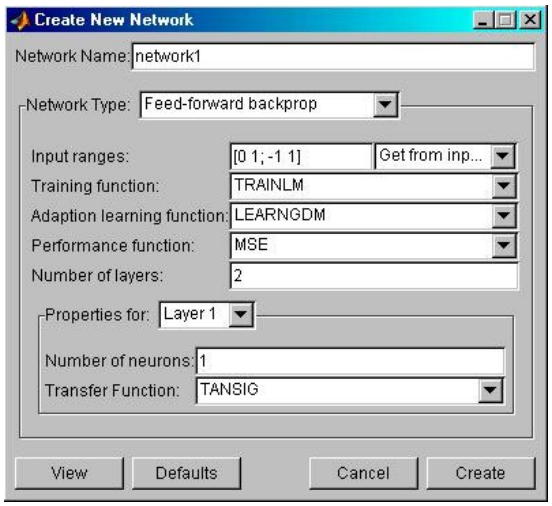


Рис. 3

Чтобы создать нейронную сеть, необходимо:

1) сформировать последовательности входов и целей (кнопка New Data) либо загрузить их из рабочей области системы MATLAB;

2) создать новую нейронную сеть (кнопка New Network) либо загрузить ее из рабочей области или из файла (кнопка Import);

3) выбрать тип нейронной сети и нажать кнопку Train, чтобы открыть окно для задания параметров процедуры обучения;

4) открыть окно Network для просмотра, инициализации, моделирования и адаптации сети.

Network Name – стандартное имя сети, присваиваемое GUI-интерфейсом, порядковый номер имени изменяется автоматически.

Network Type – тип сетей, доступных для работы в NNTool.

Inputs ranges – допустимые границы входов, которые задаются пользователем либо определяются автоматически из списка Get from Inp…

Training function – список обучающих функций.

Adaptation learning function – список функций настроек для режима адаптации.

Performance function – список функций оценки качества обучения.

Number of layers – количество слоев сети.

Properties for – свойства для слоев 1 и 2.

Numbers of neurons – количество нейронов в слое.

Transfer function – функция активации слоя.

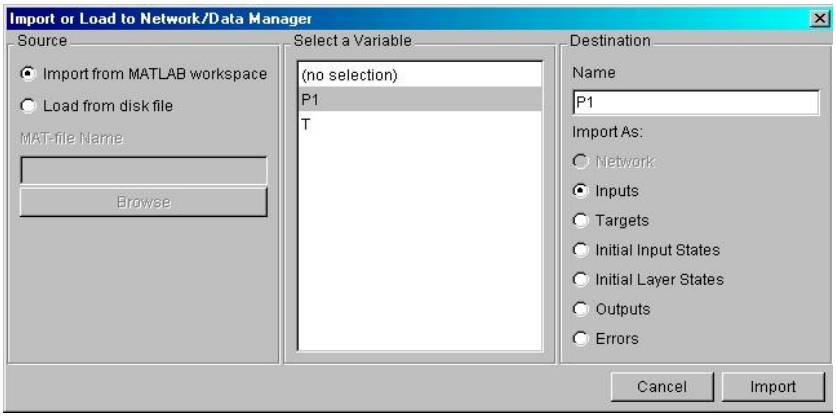


Рис. 4

Окно Import or Load to Network/ Data Manager включает три поля:

Source – поле для выбора источника данных: рабочая область MATLAB (Import from MATLAB workspace) либо файл (Load from disk file).

При выборе первого варианта поле Select a variable содержит все переменные рабочей области. При выборе второго варианта активизируется поле MAT – file Name и кнопка Browse.

Destination – поле, в котором следует указать тип импортируемых данных путем выбора одной из радиокнопок.

Окно Export or Save from Network/Data Manager (рис. 5) позволяет передать данные из рабочей области NNTool в рабочую область MATLAB или сохранить их в виде файла на диске. Если в качестве сохраняемой переменной выбрана переменная, принадлежащая классу network object, и она экспортирована в рабочую область MATLAB, можно построить модель нейронной сети в системе Simulink.

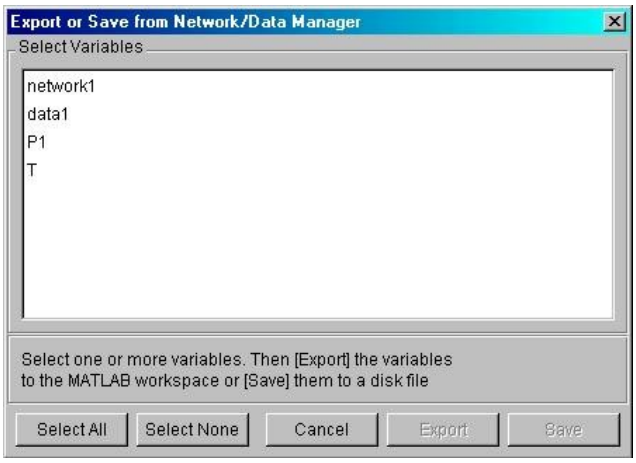
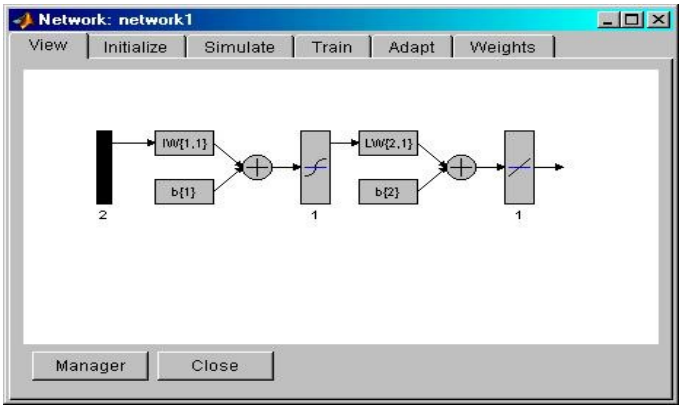


Рис. 5

После создания и активизации нейронной сети в окне Network/Data Manager становятся активными кнопки View, Delete, Initialize, Simulate, Train, Adapt. При



выделении в области Networks созданной сети открывается диалоговая панель Network: <имя_сети> (рис. 6).

Рис. 6

Панель имеет шесть закладок:

View – просмотр структуры сети.

Initialize – задание начальных весов и смещений.

Simulate – моделирование сети.

Train – обучение сети.

Adapt – адаптация и настройка параметров в сети.

Weights – просмотр установленных весов и смещений.

Особенности работы с сетью в окне Network рассмотрим на примере сетей с прямой передачей сигнала и обратным распространением ошибки.

Пример 1.1. Создать и обучить нейронную сеть выполнению операции y=x12+x2, если заданы последовательности входа P[1 0.5 0 1; -2 0 0.5 1] и цели T = [-1 0.25 0.5 2].

Решение

В окнах Create New Data следует сформировать последовательности входов и целей, а в окне Create New Network выбрать нейронную сеть типа feed-forward backprop и задать ее характеристики (см. рис. 3). Схема этой сети показана на рис. 6. Для инициализации сети следует выбрать закладку Initialize (рис. 7). В ниспадающем меню Get from input необходимо задать диапазоны значений входных данных. Для ввода исходных данных инициализации весов следует воспользоваться кнопками Set Ranges и Initialize Weights.

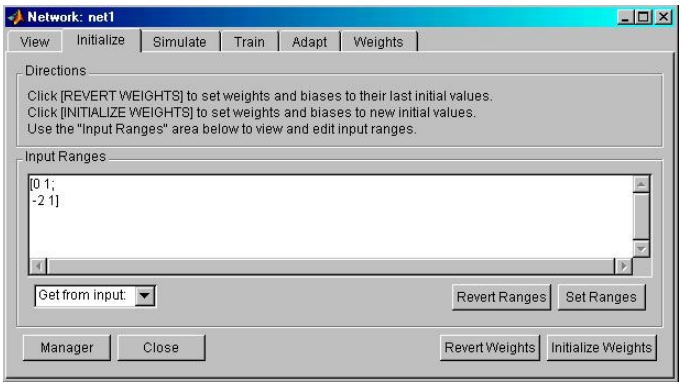


Рис. 7

Затем выполняется обучение сети, для чего выбирается закладка Train. Эта панель, в своюочередь, имеет три закладки:

Train Info – информация об обучающих последовательностях (рис. 8).

Train Parametrs – параметры обучения (рис. 9).

Optional Info – дополнительная информация (рис. 10).

Вводя значения в поля ввода и осуществляя выбор из выпадающих меню, можно установить имена последовательностей ввода T и цели P. Параметры обучения в ходе лабораторной работы изменять не рекомендуется (оставить принятые по умолчанию).

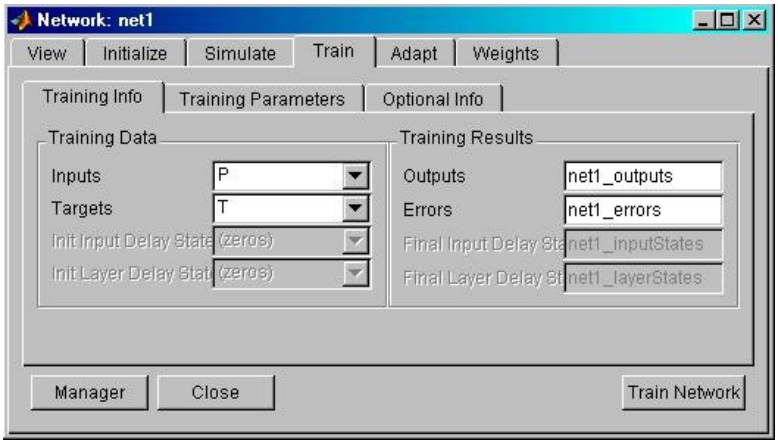
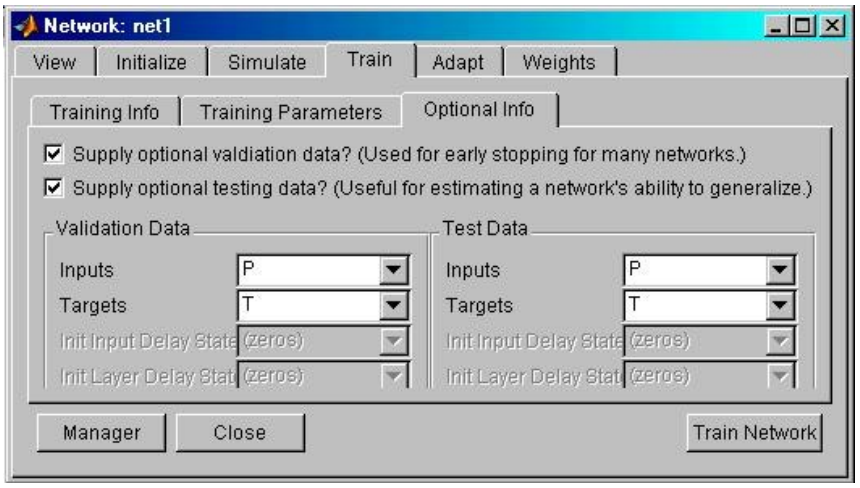


Рис. 8

Последняя закладка применяется, когда в процессе обучения используются контрольные и тестовые последовательности. Приступить к обучению сети можно путем активизации кнопки Train Network, а результаты просмотреть в окне Network/data Manager, выбрав кнопку Manager, имеющуюся на любой из шести закладок окна Network. Для этого следует активизировать переменные, соответствующие последовательностям выхода (network1_outputs) или ошибок (network1_errors), и нажать кнопку View. Качество обучения представляется в отдельном окне Train (рис. 11), где можно проследить уменьшение среднеквадратической ошибки. Значение весов и смещений можно увидеть, если выбрать закладку Weighs.



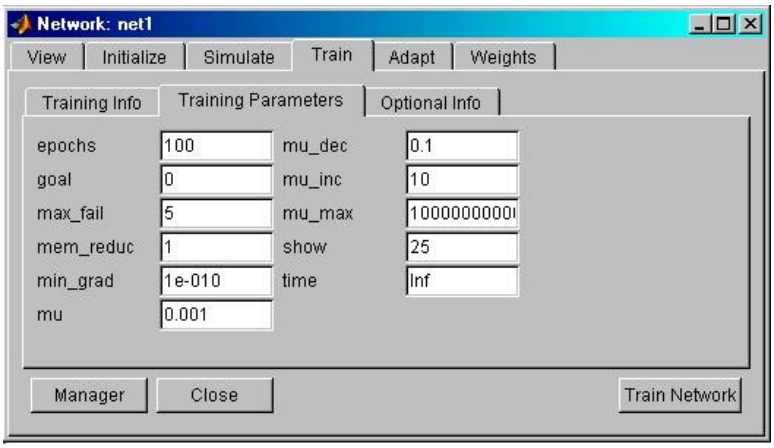


Рис. 9

Рис. 10

Для удобства работы можно экспортировать созданную нейронную сеть в рабочую область системы MATLAB и получить информацию о весах и смещениях непосредственно в рабочем окне системы:

```
>>network1.IW{1,1}, network1.b{1}
ans = -1.7530
ans = 1.1592
>>network1.LW{2.1}, network1.b{2}
```

```
ans = -1.5142
ans = 0.5025
```

Используя функцию gensim, можно построить S-модель сети в системе Simulink (рис. 12). Выбор (двойной щелчок мыши) элементов сети позволяет просматривать их структуру:

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

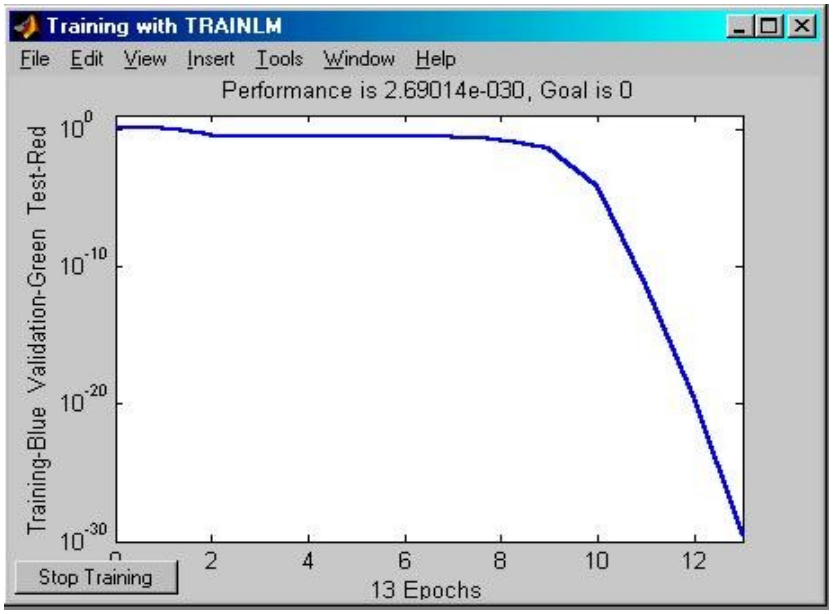


Рис. 11

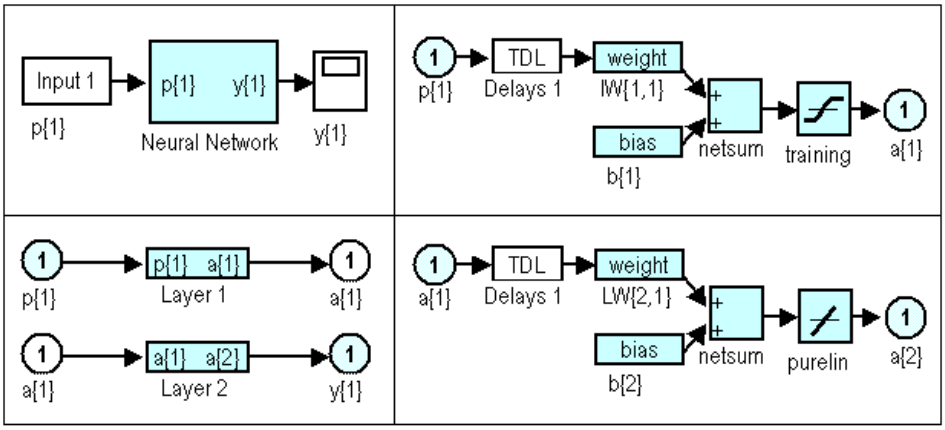


Рис. 12

Эти схемы, в отличие от иллюстративной схемы на рис. 6, являются в полной мере функциональными схемами и могут быть применены для моделирования нейронной сети.

Варианты заданий

Создать и обучить нейронную сеть для реализации функции двух переменных согласно одному из вариантов:

1) $y=x_1^2 - x_2^2+x_3$.
2) $y=x_1+x_2+x_3^2$.
3) $y=x_1^2+x_2+x_3^2$.
4) $y=x_1^2 - x_2+x_3^2$.
14) $y=x_1^2+x_2^2+x_3^2$.
15) $y=x_1^2+x_2^2 - x_3^2$.
16) $y=x_1^2 - x_2^2 - x_3^2$.
17) $y=x_1^3+x_2^3+x_3^3$.

5) $y=x_1^2+x_2-x_3^2$.
6) $y=x_1+x_2-x_3^2$.
7) $y=x_1^2-x_2^2+x_3^2$
8) $y=x_1^2+x_2^3+x_3^2$.
9) $y=x_1^3+x_2+x_3^2$.
10) $y=x_1^3+x_2^3+x_3^2$.
11) $y=-x_1+x_2-x_3^2$.
12) $y=-x_1^2-x_2^2+x_3^2$.
13) $y=-x_1^2+x_2^3+x_3^2$.
18) $y=-x_1^2+x_2+x_3^2$.
19) $y=-x_1^2-x_2^2+x_3$.
20) $y=-x_1+x_2+x_3^2$.
21) $y=-x_1^2+x_2+x_3^2$.
22) $y=-x_1^2-x_2+x_3^2$.
23) $y=-x_1^2+x_2-x_3^2$.
24) $y=-x_1^3+x_2+x_3^2$.
25) $y=-x_1^2+x_2^2+x_3^2$.
26) $y=x_1^2+x_2^2-x_3^2$.

Диапазон изменения параметров: $x_1=[-2;2]$; $x_2=[-1;1]$; $x_3=[0;4]$.

Содержание отчета

В состав отчета входят следующие разделы:

1. Краткая оценка инструментального средства.
2. Результаты создания, обучения и моделирования сети.
3. Модели сети в NNTool и Simulink.
4. Выводы.

Контрольные вопросы

1) Что представляют собой нейронные сети в среде NN Toolbox? Перечислите их основные параметры и функции.
2) Какие классы задач решаются при помощи нейронных сетей?
3) Что понимается под обучением сети?
4) Какие алгоритмы обучения нейронных сетей Вам известны?

Лабораторная работа № 3.

Программное средство Neural Network Toolbox системы MatLab.

Форма проведения: компьютерные симуляции.

Цель: Изучение свойств персептронов. Создание и обучение нейронной сети для реализации логической функции.

Порядок выполнения работы

1. Ознакомиться с демонстрационными программами, иллюстрирующими решение задачи классификации различными персептронами:

demop1 – классификация с использованием двухвходового персептрона;
demop4 – нормирование входных векторов внешнего слоя;
demop5 – обучение с использованием нормированной функции настройки;
demop6 – поведение сети при классификации линейно неразделимых векторов;
nnd3pc – классификация реальных объектов при помощи персептрона;
nnd4db – построение разделяющей линии персептрона;
nnd4pr – перемещение разделяющей линии при использовании различных функций обучения.

2. Изучить М-функции, предназначенные для инициализации, обучения и моделирования персептронов, используя справочную систему MATLAB.

3. Реализовать решение задачи классификации линейно отделимых векторов.

4. Реализовать решение задачи классификации линейно неотделимых векторов.

5. Ответить на контрольные вопросы и составить отчет.

Теоретические обоснования

Персептроны – класс линейных одно- и многослойных нейронных сетей, реализующих прямое распространение сигнала и предназначенных для решения задач классификации линейно отделимых входных векторов. Многослойные персептроны позволяют решать сложные проблемы, связанные с анализом коммутационных соединений, распознаванием образов, и другие задачи классификации с высоким быстродействием и гарантией правильного результата. Нейроны персептронов имеют ступенчатую линию активации hardlim и, следовательно, каждый нейрон возвращает единицу или ноль, что соответствует его возбуждению или торможению.

По команде help percept можно получить подробную информацию о наборе M-функций, относящихся к построению нейронных сетей на основе персептронов. Структура персептрона представлена на рис. 13.

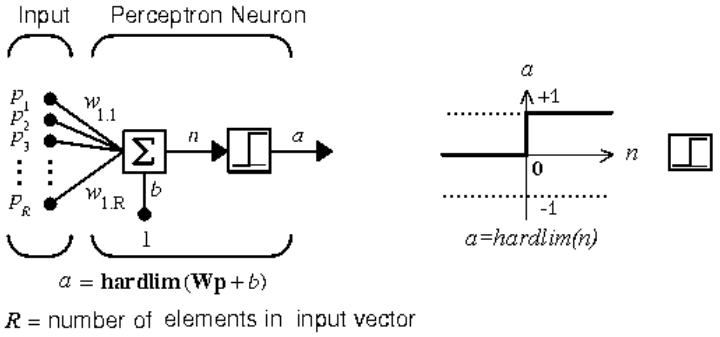


Рис. 13

Для формирования модели однослойного персептрона предназначена функция newp:

net=newp(PR, S, tf, lf),

где net – имя объекта "Нейронная сеть"; PR – массив min и max значений для R элементов входа размером R*2; S – число нейронов в слое ; tf – функция активации {hardlim или hardlims}; lf – обучающая функция {learnp или learnpn}. При кратком вызове net=newp(PR, S) по умолчанию будут использоваться функции hardlim и learnp. Например, вызов функции с параметрами ([-2 2;-3 3], 1) создает персептрон с двухэлементным входом и одним нейроном, значения элементов входного вектора созданной сети должны варьироваться в диапазонах [-2, 2] и [-3, 3]. Просмотреть различные параметры и значения персептрона можно следующим образом:

```
>>inputweight = net.inputweights{1, 1}% параметры входов
>>biases=net.biases{1}        % параметры смещений
>>net          % параметры и архитектура сети
>>net.IW{1}, net.b{1}% значения весов и смещений
```

Для того чтобы изменить нулевые значения весов и смещений, установленные по умолчанию, можно применить следующие операторы:

```
>>net.IW{1}=[-1 1];  net.b{1}=[1];
```

Вернуться к первоначальным установкам параметров персептрона можно с помощью функции init :

```
>>net=init(net);
```

Можно изменить способ, которым инициализируется персептрон, с помощью функции init. Для этого достаточно изменить типы функций инициализации, которые применяются для установки первоначальных значений весов входов и смещений. Например, чтобы задать случайные значения параметрам персептрона, следует воспользоваться функцией rands:

```
>>net.inputweigts{1, 1}.initFcn='rands';
>>net.biases{1}.initFcn='rands';
>>net=init(net);
>>wts=net.IW{1, 1}
wts=  -0,96211      0,3526    % возможные значения
>>bias=net.b{1}
b= -0,032452                  % возможное значение
```

Еще один способ задания сети, который использует определение массива элементов входа:

```
>>PR = [-1 1; -2 2; -3 3]     % матрица 2×3 (2 – min и max
   значения для 3 входов нейрона)
>>net1=newp(PR, 2)  % создание сети с 2 нейронами
```

Адаптация и обучение персептронов выполняются 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 функцией mal. Применение функции адаптации гарантирует, что любая задача классификации с линейно отделимыми векторами будет решена за конечное число циклов (шагов настройки). Функция адаптации корректирует параметры сети по результатам обработки каждого входного вектора. Эта функция относится к категории устаревших, однако для сохранения преемственности с предшествующими версиями NNT ее использование допустимо. Новая обучающая функция train реализует адаптивное обучение с последовательным представлением входов, при котором настройка сети выполняется не после каждого прохода, а в результате всех проходов обучающего множества. К сожалению, не существует доказательства, что такой алгоритм обучения персептрона является сходящимся. Поэтому использование функции adapt для персептрона является предпочтительным.

Пример 2.1. Обучение персептрона выполнению функции логическое И.

Решение

Определим последовательность двухэлементных векторов входа P для сети с одним двухвходовым нейроном:

```
>> P=[0 0 1 1 ; 0 1 0 1] % матрица 4×2
```

Сформируем последовательность целей T:

```
>>T = [0  0  0  1]        % матрица 4×1
```

Установим число проходов равным 10 и выполним адаптацию:

```
>>net.adaptParam.passes=10;
>>net=adapt(net, P, T);
```

Просмотрим значения весов и смещения, отобразим векторы P и T и построим график разделяющей линии:

```
>>net.IW{1}, net.b{1}
ans= 2       1      % возможные значения
ans= -3             % возможные значения
>>plotpv(P, T);
>>linehandle=plotpc(net.IW{1}, net.b{1});
```

Промоделировать спроектированную нейронную сеть можно с помощью функции sim, подав на вход сети обучающую последовательность, совпадающую с входной последовательностью P:

```
>>Y=sim(net, P)
Y=0  0  0  1   % настройка сети выполнена правильно
```

Длительность обучения персептронов весьма чувствительна к разбросу значений длины отдельных векторов. Уравновесить влияние больших и малых значений входных компонент можно путем масштабирования вектора входа. Нормированное правило обучения персептронов реализуется функцией learnpn:

```
>>net.inputweights{1, 1}.learnFcn:='learnpn';
>>net=init(net);
```

Для решения более сложных задач можно использовать сети с бóльшим числом нейронов. Например, для классификации входных векторов по 4 группам можно построить сеть с двумя нейронами, чтобы сформировать две разделяющее линии и приписать каждому вектору свою область. Получить S-модель сети в Simulink можно путем вызова уже известной функции gensim.

Варианты заданий

Создать и обучить нейронную сеть для реализации логической функции согласно одному из вариантов:

1. $(\mathrm{A} \vee \mathrm{B}) \Rightarrow (\mathrm{A} \wedge \mathrm{B})$
2. $\overline{(\mathrm{A} \vee \mathrm{B}) \vee \overline{\mathrm{A}}}$
3. $\overline{(\mathrm{A} \vee \mathrm{B}) \vee (\mathrm{B} \Leftrightarrow \mathrm{A})}$
4. $\overline{\mathrm{A} \Rightarrow \mathrm{B}} \vee (\overline{\mathrm{A}} \wedge \mathrm{B})$
5. $(\overline{\mathrm{A}} \Leftrightarrow \mathrm{B}) \wedge \overline{\mathrm{A} \wedge \mathrm{B}}$
6. $\left|(A \Rightarrow B) \wedge B\right| \Rightarrow A$
7. $(A \Rightarrow B) \vee A \Rightarrow B$
8. $\overline{B} \wedge (A \Rightarrow B) \Rightarrow \overline{A}$
9. $(A \Rightarrow B) \Leftrightarrow (\overline{B} \Rightarrow \overline{A})$
10. $\overline{A} \wedge (A \vee B) \Rightarrow B$
11. $\overline{A} \wedge B \vee (\overline{A \vee B})$
12. $A \Rightarrow (\overline{A} \vee \overline{B})$
13. $A \wedge B \Rightarrow (\overline{A \vee \overline{B}})$
14. $(A \wedge B) \Leftrightarrow (\overline{A \vee B})$
15. $(\overline{A} \vee \overline{B}) \Leftrightarrow B$
16. $(A \Rightarrow B) \Rightarrow (A \wedge B)$
17. $(\overline{A} \Rightarrow \overline{B}) \wedge (A \vee B)$
18. $A \wedge (A \Rightarrow B) \vee \overline{B}$
19. $(A \vee B) \wedge \overline{B}$
20. $(A \Leftrightarrow B) \wedge (\overline{A} \wedge B)$
21. $(A \Rightarrow B) \vee (\overline{A} \wedge B)$
22. $(A \vee B) \Leftrightarrow (\overline{A} \wedge B)$
23. $(A \Leftrightarrow B) \wedge (\overline{A} \vee B)$
24. $(A \Leftrightarrow B) \wedge (A \wedge \overline{B})$
25. $(A \Leftrightarrow \overline{B}) \vee (\overline{A} \wedge B)$
26. $(A \wedge B) \wedge (\overline{A} \wedge B)$
27. $(A \vee B) \wedge (\overline{A} \wedge \overline{B})$
28. $(A \vee B) \wedge \overline{(A \wedge B)}$
29. $(A \wedge B) \vee \overline{(A \wedge B)}$
30. $\overline{(A \wedge B)} \vee (A \wedge B)$

Для определения функции цели можно воспользоваться таблицей:

A	B	$\overline{A}$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Отчет по выполненной работе должен содержать следующие разделы:

1) Краткое описание инструментального средства NNTool.
2) Краткая характеристика персептронов.
3) Последовательность команд для создания, инициализации и обучения сети.
4) Структура сети.
5) График изменения параметров и величина ошибки в процессе обучения.
6) график разделяющей линии.

Контрольные вопросы

1. Какие способы реализации нейронных сетей Вам известны?
2. Каковы особенности нейронных сетей типа персептрон?
3. Опишите классический алгоритм обучения персептрона.
4. Что понимается под архитектурой нейронной сети?
5. Что понимается под разделяющей линией и разделяющей плоскостью персептрона?
6. Какими способами можно произвести настройку параметров персептрона? В чем их отличие?
7. В каком случае персептроны могут решать линейно неотделимые задачи?

Лабораторная работа № 4.

Персептрон. Линейные нейронные сети. Радиальные базисные сети.
Форма проведения: компьютерные симуляции (2 часа)
Цель: Исследование сетей прямого распространения сигнала

Порядок выполнения работы

1. Изучить демонстрационные примеры, иллюстрирующие работу сетей с прямой передачей сигнала:

nnd11nf – функционирование сети;
nnd11bc – демонстрация алгоритма обратного распространения ошибки;
nnd11fa – функция аппроксимации;
nnd11gn – обобщающие способности сети;
appcs1 – распознавание нелинейной системы;
appcs2 – линеаризация с помощью сети с обратным распространением ошибки.

2. С помощью справочной системы MATLAB ознакомиться с функциями, предназначенными для работы с указанными сетями.
3. Спроектировать несколько вариантов архитектуры сети для решения задачи распознавания образов, закодированных с помощью матрицы $N \times M$ и предусмотренных вариантом.
4. Создать, обучить и промоделировать сеть на идеальных и зашумленных входных векторах, построить графики процесса обучения и результатов моделирования.

5. Выбрать сеть с наилучшей архитектурой.
6. Пункты 3 – 5 повторить для решения задач регрессии и прогнозирования.
7. Ответить на контрольные вопросы, составить отчет.

Теоретические обоснования

В сети с прямой передачей сигнала нейроны регулярным образом организованы в слои. Каждый нейрон внутренних (скрытых) слоев и выходного слоя соединен со всеми элементами предыдущего слоя. Такие сети – с полной системой связи – предпочтительны для большинства приложений в отличие от сетей с выборочными связями. Архитектура однослойной сети с S нейронами, логистической функцией активации и R входами представлена на рис. 14 (а – детальное представление; б – нотация MATLAB). Архитектура сети с одним скрытым слоем представлена на рис. 15.

Сети с прямым распространением сигнала используются для решения задач классификации либо выступают в качестве идеального аппроксиматора, т.е. они могут моделировать функцию практически любой сложности (задачи регрессии, прогнозирования). Число слоев и число элементов в каждом слое зависят от сложности функции и определяются эмпирически. При этом следует учитывать, что если более сложная модель не дает результат лучше, чем более простая, то из этих двух моделей следует предпочесть последнюю. В качестве начального приближения можно взять сеть с одним промежуточным слоем, а число элементов в нем положить равным полусумме числа входных и выходных элементов.

По команде help backprop можно получить подробную информацию о наборе М-функций, относящихся к построению нейронных сетей с прямой передачей сигнала. Количество входных и выходных элементов определяется условиями задачи. Входные переменные могут выбираться исследователем интуитивно, но все они должны быть значимыми.

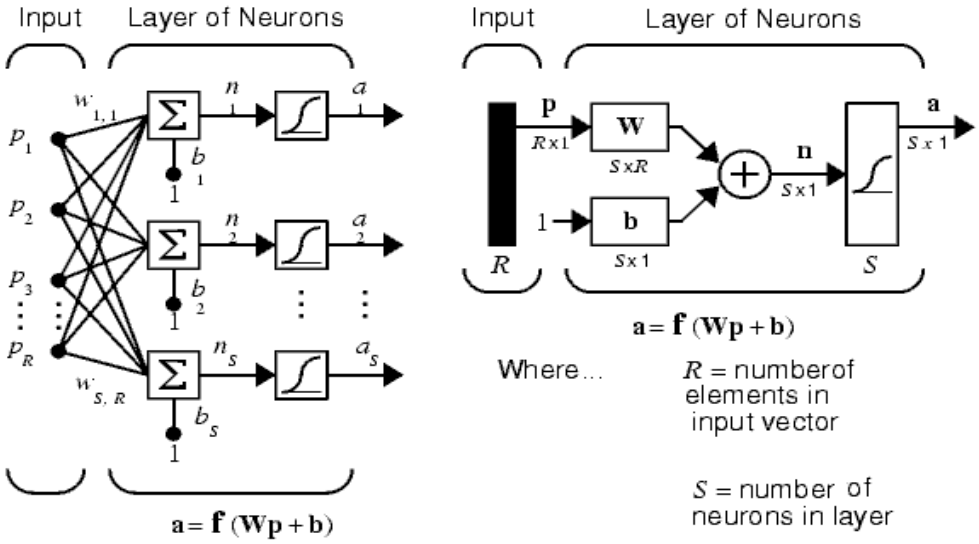


Рис. 14 (а, б)

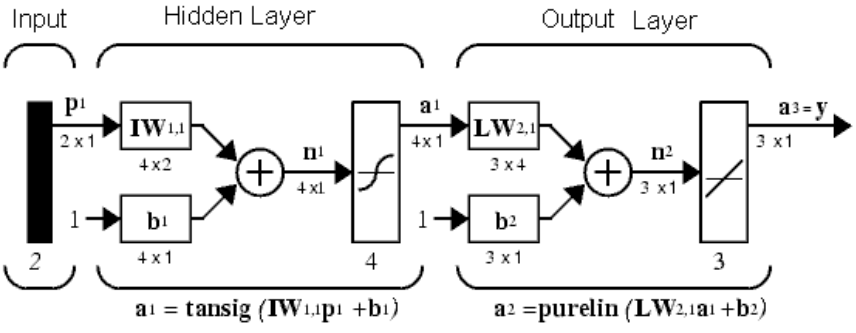


Рис. 15

Для создания сети прямого распространения используется функция newff:

net = newff(PR, [S_1 S_2...S_{Nl}], {tf1 tf2...tfnl}, btf, blf, pf),

где PR – матрица Rx2 минимальных и максимальных значений для R входных элементов; Si – размер i-го слоя для Nl слоев; tfi – функция активаци i-го слоя (по умолчанию tansig); btf – обучающая функция (trainlm); blf – обучающая функция для настройки весов и смещений (learngdm); pf – характеристическая функция (mse).

Пример 3.1. Создать нейронную сеть для распознавания 26 символов латинского алфавита на основе оцифровки каждого символа. Проектируемая сеть должна точно распознавать идеальные векторы входа и с максимальной точностью воспроизводить распознавание зашумленных векторов. Правильно функционирующая сеть должна ответить вектором со значением единица для элемента, соответствующего номеру символа в алфавите, и ноль для всех остальных символов.

Решение

Каждый символ (образ) может быть представлен шаблоном 5×7 (рис. 16), 10×14 и др. с помощью стандартной функции MATLAB prprob. Она определяет 26 векторов входа, каждый из которых содержит 35 элементов (массив латинского алфавита). Функция prprob формирует выходные переменные alphabet и targets, которые соответственно определяют массивы алфавита и целевых векторов.

		X		
	X		X	
	X		X	
X				X
X	X	X	X	X
X				X
X				X

Шаблон символа ‘A’

0	0	1	0	0
0	1	0	1	0
0	1	0	1	0
1	0	0	0	1
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1

Вектор входа символа ‘A’

Рис. 16

Сформируем массив векторов входа alphabet 35×26 с шаблонами символов алфавита и массив целевых векторов targets:

```
>>[alphsbet, targets]=prprob;
>>[R, Q]=size(alphabet);
>>[S2, Q]=size(targets);
```

Число нейронов скрытого слоя S1 рекомендуется определить эмпирическим путем. Если при обучении сети возникнут затруднения, то можно увеличить количество нейронов этого слоя. Создадим двухслойную сеть с сигмоидальными функциями активации нейронов:

```
>>net=newff(minmax(alphabet),[S1,S2],{'logsig',  'logsig'},
  'traingdx');
>>net.LW{2, 1}=net.LW{2, 1}*0.01
>>net.b{2}=net.b{2}*0.01
```

Первоначально обучим сеть в отсутствии шума с заданным числом циклов либо до достижения допустимой квадратичной погрешности:

```
>>P= alphabet;
>>T=targets;
>>net.performFcn='sse';
>>net.trainParam.goal=0.1;    %    допустимая    погрешность
  обучения
>>net.trainParam.show=20; % интервал выхода информации
>>net.trainParam.epochs=5000; % количество эпох обучения
>>net.trainParam.mc=0.95;     % параметр возмущения
```

```
>>[net,tr]=train(net, P, T)
```

Для проектирования сети, не чувствительной к воздействию шума, можно обучить ее на 10 наборах двух идеальных и двух зашумленных копий векторов алфавита:

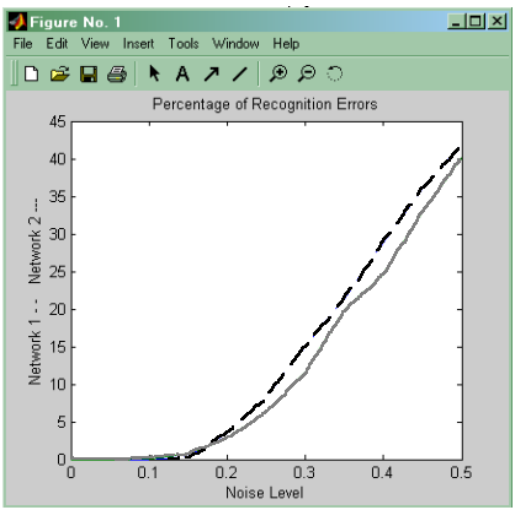
```
>>netn=net;
>>netn.trainParam.goal=0.6;
>>netn.trainParam.epochs=300;
>>T=[targets targets targets targets];
>>for pass=1:10
>>P=[alphabet, alphabet, (alphabet + randn(R, Q)*0.1),
  (alphabet+randn(R, Q)*0.2)];
>>[net,tr]=train(netn, P, T);
>>end
```

Так как нейронная сеть обучалась в присутствии шума, она потеряла способность распознавать идеальные векторы. Обучим сеть идеальным векторам еще раз:

```
>>net.trainParam.goal=0.1;
>>net.trainParam.epochs=500;
>>net.trainParam.show=5;
>>[net,tr]=train(netn, P, T);
```

Осуществим проверку на 100 векторах входа при различных уровнях шума. Рассмотрим две сети: network1 – сеть, обученную на идеальных векторах, и network2 – сеть, обученную на зашумленных последовательностях. Наличие шумов может привести к тому, что сеть не будет формировать векторы выхода, состоящего точно из 1 и 0. Поэтому используется функция compet, которая присваивает 1 единственному элементу вектора выхода, а всем остальным – значение 0:

```
>>noise_range=0 : .05 : .5;
>>max_test=100;
>>network1=[ ];
>>network2=[ ];
>>T=targets;
Протестируем сети:
>>for noiselevel=noise_range
     errors1=0;
     errors2=0;
     for i=1: max_test
          P= alphabet+randn(35, 26)*noiselevel;
          A=sim(net,P);            % тест для сети 1
          AA=compet(A);
          errors1=errors1+sum(sum(abs(AA-T)))/2;
          B=sim(netn, P);              % тест для сети 2
     BB=compet(B);
          errors2=errors2+sum(sum(abs(BB-T)))/2;
          echo off
     end
```



```
>>network1=[network1 errors1/26/100]; % средние значения
>>network2=[network2 errors2/26/100];   % ошибок
>>end
```

Построим графики зависимости погрешности от уровня входного шума (рис. 17):

```
>>plot(noise_range, network1*100, '--', noise_range,
network2*100);
```

Проверим работу сети на конкретном символе ‘T’. Для этого сформируем зашумленный вектор входа для этого символа (рис. 18), промоделируем сеть и определим соответствующий идеальный символ (рис. 19):

```
>>masa=alphabet(:, 20)+randn(35, 1)*0.2;
>>plotchar(masa);
```

Рис. 17

```
>>A2=sim(net, masa);
>>A2=compet(A2);
>>otvet=find(compet(A2)==1)
>>plotchar(alphabet(:, otvet))
```

Рис. 18

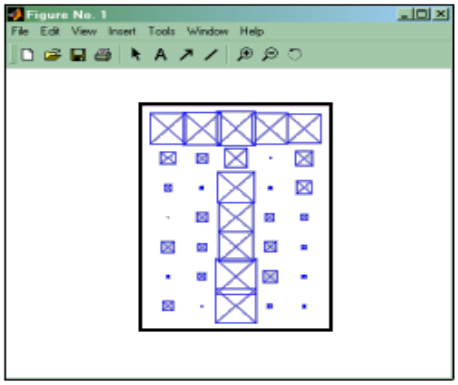
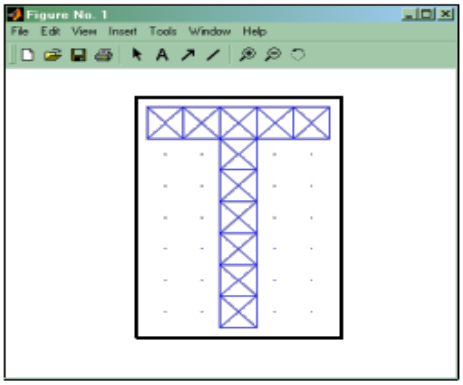


Рис. 19



Если необходима более высокая точность распознавания, сеть может быть обучена либо в течение более длительного времени, либо с использованием большего числа нейронов в скрытом слое. Можно увеличить размер векторов, чтобы использовать шаблон с более мелкой сеткой.

Задачи прогнозирования (анализа временных рядов) являются частным случаем регрессии. Причем можно строить прогнозы как для одной, так и для нескольких переменных как на один, так и на несколько шагов вперед. Такие задачи предполагают создание (обучение) сетей, способных обнаружить скрытые закономерности в последовательности чисел и предсказать следующие (будущие) значения переменных.

Варианты заданий

Задачи классификации

Создать нейронную сеть для распознавания N символов кириллицы на основе оцифровки каждого символа (символ может быть представлен как шаблоном 5×7, так и шаблоном 10×14), содержащую:

1) неповторяющиеся буквы имени и фамилии, 5×7;
2) неповторяющиеся буквы имени и фамилии, 10×14;
3) 10 строчных букв греческого алфавита, 5×7;
4) 10 прописных букв греческого алфавита, 10×14;
5) 10 символов шрифта Webdings, 10×14;

6) 10 арабских цифр, 10×14.

Варианты с 7 по 20 задаются преподавателем.

Содержание отчета

В состав отчета входят следующие разделы:

1. Формулировка задачи.
2. Анализ demo-примера (по указанию преподавателя).
3. Архитектура и параметры сети; функция для формирования образов; сценарии обучения и результаты моделирования для задач классификации и прогнозирования.
4. Выводы.

Контрольные вопросы

1. В чем заключается проблема переобучения сети?
2. Что понимается под обобщающей способностью нейронной сети?
3. Сформулируйте основные принципы метода обратного распространения? Какие модификации этого метода Вам известны?
4. Как вычисляется ошибка для сети с многослойной архитектурой?
5. Что понимается под термином «поверхность ошибок»?

Лабораторная работа № 5.

Самоорганизующиеся и рекуррентные сети. Гибридные нейронные сети.
Форма проведения: компьютерные симуляции.
Цель: Исследование самоорганизующихся нейронных сетей Кохонена.

Порядок выполнения работы.

1. Изучить демонстрационные примеры, иллюстрирующие работу слоев и сетей Кохонена:

   democ1 – настройки слоя Кохонена;
   demosm1 – одномерная карта Кохонена;
   demosm2 – двумерная карта Кохонена.
2. Изучить М-функции, предназначенные для инициализации, обучения и моделирования линейных сетей, используя справочную систему MATLAB.
3. Получить задание у преподавателя и реализовать его решение при помощи линейной сети.
4. Ответить на контрольные вопросы и составить отчет.

Теоретические обоснования

В процессе анализа больших информационных массивов неизменно возникают задачи, связанные с исследованием структуры данных, их объединением в группы

(кластеры), распределением по классам, сжатием данных, определением характеристик объекта по ограниченному набору признаков и т.д. Такие задачи успешно решаются с помощью самоорганизующихся сетей. Сетям этого типа в процессе обучения не нужен "учитель", т.е. в процессе тренировки не требуется сообщать сети правильный ответ при предъявлении примера (входного образа). Эти сети способны осуществлять кластеризацию, т.е. разделение данных на различные категории (кластеры). Классическим примером таких сетей являются так называемые карты Кохонена – нейронные сети, в которых близко расположенные нейроны соответствуют близким кластерам (упорядочены). Архитектура сети Кохонена представлена на рис. 20. Обращаем внимание, что в нейронах сети не используются смещения. По команде help selforg можно получить информацию об М-функциях, относящихся к построению карт Кохонена.

Создание самоорганизующейся карты Кохонена осуществляется функцией newsom:

net = newsom(PR, [D1,D2,...], tfcn, dfcn, olr, osteps, tlr, tns),

где D_i – размерность карты (по умолчанию = [5 8]); tfcn – топология ('hextop'); dfcn – функция вычисления дистанционного расстояния между нейронами ('linkdist'); olr – скорость фазы упорядочения коэффициентов (0.9); osteps – шаг фазы упорядочения (1000); tlr – скорость этапа подстройки коэффициентов (0.02); tnd – начальные окрестности этапа подстройки (1).

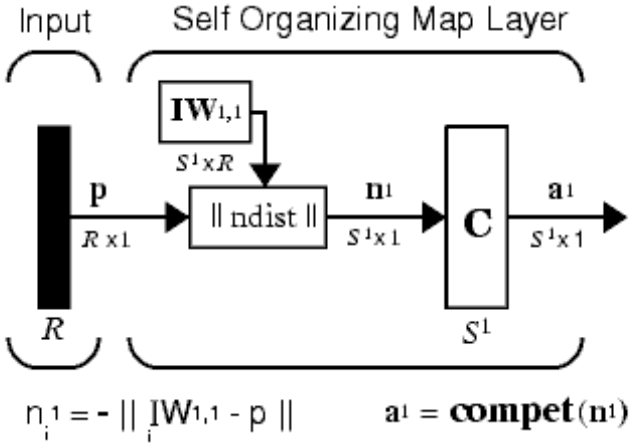


Рис. 20

Обучение сети реализуется повекторно независимо от того, выполняется ли оно с помощью функции trainwb1 или adaptwb. В обоих случаях функция learnsom выполняет настройку весовых коэффициентов нейронов. При обучении на карте Кохонена одновременно изменяются весовые коэффициенты нейронов, соседствующих с нейроном-победителем в пределах некоторого радиуса (окрестности). Пример окрестности радиуса 1 и 2 представлен на рис. 21.

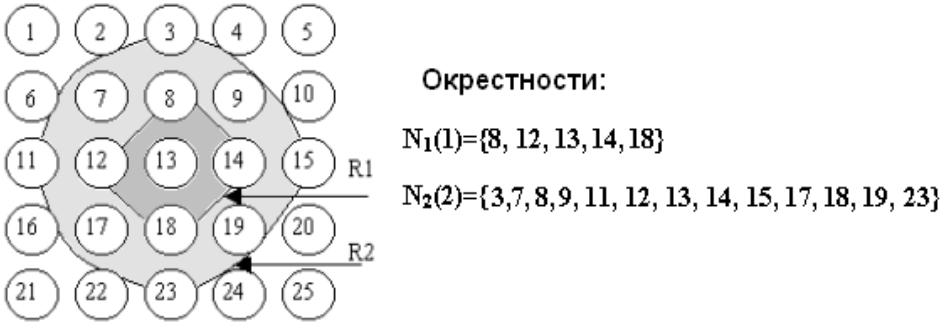


Рис. 21

Карта Кохонена может быть не только двухмерной, но и одномерной, трехмерной и бóльших размерностей.

Пример 4.1. Создание карт Кохонена различной топологии.

Решение

```
>>pos=gridtop (2,3)% создание прямоугольной сетки 2x3
pos =   0 1 0 1 0 1
        0 0 1 1 2 2
>>plotsom(pos)      % отображение сети
>>pos = gridtop(3, 2)    % создание прямоугольной сетки 3x2
pos =   0 1 2 0 1 2
         0 0 0 1 1 1
>>pos=hextop (2, 3);% создание гексагональной сетки,
% используется по умолчанию
pos= 0    1.000     0.5000    1.5000    0.000     1.000
     0    0.000     0.866     0.866     1.732     1.732
>>pos=randtop (2, 3);     % создание сети
                  % со случайным расположением узлов
pos=0.061 0.6471    0.408     0.949     0.000     1.651
    0.000 0.122     0.904     0.547     1.401     1.568
```

Вычисление расстояний между нейронами осуществляется четырьмя вышеупомянутыми функциями:

```
>>pos=gridtop(2, 3);% пример для прямоугольной сетки
>>d=dist(pos)
>>d=linkdist(pos)   % используется по умолчанию
>>d=mandist(pos)
```

Пример 4.2. Разделение набора входных векторов с использованием графики MATLAB.

Решение

Сформируем 48 координат случайных точек:

```
>>c=8;          % число кластеров
>>n=6;          % число векторов в кластере
>>t=c*n;  % число точек
>>d=0.5;         % среднеквадратичное отклонение от центра в
                 кластере
>>x=[-10 10; -5 5]; % диапазон входных значений
>>[r, q]=size(x); minv=min(x')';maxv=max(x')';
>>v=rand(r,c).*((maxv - minv)*ones(1,c)+minv*ones(1,c));
>>v=[v v v v v v]; v=v+randn(r, t)*d;   % координаты точек
>>P=v;
>>plot(P(1, :), P(2, :), '+k') ;   %отображение     векторов
  входа
```

Создадим сеть из 8 нейронов:

```
>>net=newsom([-2 12; -1 6], [2, 4]);
>>w0=net.IW{1}
>>b0=net.b{1}
>>c0=exp(1)./b0% параметр активности нейронов
Обучим сеть:
>>net.trainParam.epochs=500;
>>net=train(net, P);
>>w=net.IW{1}
>>bn=net.b{1}
>>cn=exp(1)./bn
```

Промоделируем сеть на входных векторах и просмотрим результат:

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

```
>>bar(sum(a'))  % распределение векторов по кластерам.
```

Пример 4.3. Создание и обучение сети Кохонена для обработки двухэлементных векторов.

Решение

```
>>net =newsom([0 2; 0 2], [2 3]);  % инициализация сети
>>net.layers{1}  % получение информации о сети
```

Зададим входной вектор из 12 элементов:

```
>>p=[0.1 0.3 1.2 1.1 1.8 1.7 0.1 0.3 1.2 1.1 1.8 1.7;0.2 0.1
      0.3 0.1  0.3 0.2 1.8 1.8 1.9 1.9 1.7 1.8];
>>plotsom (net.IW{1, 1}, net.layers{1}.distances)
>>hold on
Отобразим начальное положения нейронов на графике
>>plot(p(1, :), p(2, :),'*k', 'markersize', 10);
Зададим параметры обучения:
>>net.trainParam.epochs=2000;
>>net.trainParam.show=100;
>>net=train(net, p);
>>plot(p(1, :), p(2, :), '*', 'markersize', 10);
>>hold on
>>plotsom(net.IW{1, 1}, net.layers{1}.distances)
% моделирование сети на массиве обучающих векторов
>>a=sim(net, p)
a =   (2,1)  % векторы входов 1 и 2 отнесены к кластеру №2
(2,2)
(1,3)  % векторы  3 – 6 отнесены к кластеру №1
(1,4)
(1,5)
(1,6)
(4,7)  % векторы  7,8 отнесены к кластеру №4
(4,8)
(6,9)  % векторы  9,10 отнесены к кластеру №6
(6,10)
(5,11)  % векторы  11, 12 отнесены к кластеру №5
(5,12)
>>bar(sum(a')) % распределение векторов по кластерам.
```

Процесс обучения карты Кохонена включает 2 этапа: упорядочение весовых коэффициентов и подстройка, в результате чего осуществляются изменения размера окрестности и скорости обучения. Начальный размер окрестности равен максимальному расстоянию между нейронами и уменьшается до величины net.inputWeight{1,1}. learnParam.tune_nd, которая по умолчанию равна 1. Это происходит на этапе упорядочения за фиксированное количество циклов (по умолчанию 1000). Число шагов на этапе подстройки значительно превышает число шагов на этапе разложения и здесь происходит более тонкая настройка при малых окрестностях и медленном уменьшении параметра скорости обучения. Таким образом, при обучении карты Кохонена решается не только задача кластеризации, но и выполняется частичная классификация.

Слой нейронов карты Кохонена можно представить в виде гибкой сетки, которая натянута на пространство входных векторов. Из-за того, что в процессе обучения участвуют соседи нейрона-победителя, топологическая карта выглядит упорядоченной.

Варианты заданий

1. Собрать информацию о студентах группы:
   физические параметры (пол, рост, вес, цвет глаз…);

учебные результаты (успеваемость по группам дисциплин, по отдельным дисциплинам, средние баллы, …);

психологические особенности (темперамент, результаты психологического тестирования, …).

Провести анализ объектов, используя различные сочетания собранных данных. Проанализировать полученные результаты, попытаться определить, какой смысл имеют выявленные кластеры.

2. Провести классификацию ряда элементов периодической системы Д.И. Менделеева.
3. Провести классификацию ряда государств Европы, Азии, Латинской Америки.
4. Провести классификацию футбольных клубов/игроков Европы.
5. Провести классификацию легковых (грузовых) автомобилей.
6. Провести классификацию учреждений высшего образования (необходимо определить вид высшего учебного заведения: университет, академия, институт).

Содержание отчета

В состав отчета входят следующие разделы:

1. Формулировка задачи.
2. Анализ demo-примеров (по указанию преподавателя).
3. Архитектура и параметры слоя Кохонена, входной вектор.
4. Сценарий обучения слоя Кохонена, результаты моделирования.
5. Пункты 2 – 5, выполненные для карты Кохонена.
6. Выводы.

Контрольные вопросы

1. В чем заключается свойство самоорганизации нейронных сетей?
2. Приведите конкретные примеры из собственного опыта, где можно было бы использовать самоорганизующиеся сети.
3. Как происходит обучение карты Кохонена?
4. Какую роль играет параметр активности нейрона в сетях Кохонена?

КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Оценка «отлично» выставляется студенту, если глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.

Оценка «неудовлетворительно» выставляется студенту, если неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

Текущая аттестация студентов проводится преподавателями, ведущими лабораторные занятия по дисциплине, в следующих формах: отчет, собеседование.

Защита отчета проходит в форме доклада студента по выполненной работе и ответов на вопросы преподавателя.

Максимальное количество баллов студент получает, если оформление отчета соответсвует установленным требованиям, а отчет полностью раскрывает суть работы. Основанием для снижением оценки являются:

- частично не соовествует установленным требованиям;
- в отчете непольностью раскрывается суть работы.

Отчет может быть отправлен на доработку в следующих случаях:

- польностью не соовествует установленным требованиям;
- не раскрыта суть работы.

Критерии оценивания отчетов приведены в Фонде оценочных средств по дисциплине «Теория нейронных сетей».

Результатом итоговой проверки знаний студентов по дисциплине учебным планом предусмотрен зачет с оценкой (дифференцированный зачет), который выражается в виде оценок «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

10.1. Рекомендуемая литература

10.1.1. Основная литература:

1. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов. – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2013. – 400 с.
2. Медведев В.С., Потемкин В.Г. Нейронные сети. MATLAB 6 / Под общ. ред. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2012. – 496 с.

10.1.2. Дополнительная литература:

1. Галушкин А.И. Теория нейронных сетей. – М.: ИПРЖР, 2000. – 416 с.
2. Хайкин С. Нейронные сети. Полный курс. – М.: Изд. дом Вильямс, 2006.
3. Чернышев А.Б., Козлов В.А., Жерносек И.А. Нейрокомпьютерные сети: Лабораторный практикум. – Пятигорск: Изд-во ПГТУ, 2011. – 40 с.

10.1.3. Методическая литература:

1. Методические указания по выполнению лабораторных работ по дисциплине «Теория нейронных сетей».
2. Методические указания по выполнению контрольной работы по дисциплине «Теория нейронных сетей».
3. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Теория нейронных сетей».

10.1.4. Интернет-ресурсы:

1. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
2. http://window.edu.ru – образовательные ресурсы ведущих вузов

10.1.5. Программное обеспечение

1. Neural Network Toolbox системы MatLab.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

для обучающихся по организации и проведению самостоятельной работы по дисциплине «ТЕОРИЯ НЕЙРОННЫХ СЕТЕЙ» для студентов направления подготовки **09.04.02 Информационные системы и технологии** направленность (профиль) **Технологии работы с данными и знаниями, анализ информации**

Пятигорск
2024

СОДЕРЖАНИЕ

1. ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ....35
3. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ....35
4. КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ....55
9. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ....55

1. ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью освоения дисциплины «Теория нейронных сетей» является овладение необходимыми теоретическими знаниями и практическими навыками в области современных методов и подходов параллельной обработки информации для задач построения современных информационных сетей и систем.

Задачами курса являются: знакомство с моделями операционного блока нейрокомпьютера; изучение основных моделей нейронных сетей; изучение основ конструирования и настроек нейронных сетей; применение методов обучения нейронных сетей; изучение программной эмуляции нейрокомпьютеров и нейронных сетей.

2. ТЕХНОЛОГИЧЕСКАЯ КАРТА САМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТА

Для студентов очной формы обучения:

Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Обьем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
Подготовка к лекциям	конспект	Собеседование	1,62	0,18	1,8
Самостоятельное изучение литературы	конспект	Собеседование	49,32	5,48	54,8
Подготовка к лабораторным работам	отчет	Отчет письменный	4,86	0,54	5,4
Выполнение контрольной работы	контрольная работа	Контрольная работа	9	1	10
		Итого	**64,8**	**7,2**	**72**

Для студентов заочной формы обучения:

Вид деятельности студентов	Итоговый продукт самостоятельной работы	Средства и технологии оценки	Обьем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
Подготовка к лекциям	конспект	Собеседование	0,36	0,04	0,4
Самостоятельное изучение литературы	конспект	Собеседование	79,56	8,84	88,4
Подготовка к лабораторным работам	отчет	Отчет письменный	1,08	0,12	1,2
Выполнение контрольной работы	контрольная работа	Контрольная работа	9	1	10
		Итого	**90**	**10**	**100**

3. СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Тема самостоятельного изучения № 2.
Методы обучения нейронных сетей. Архитектура нейронных сетей.

Вид деятельности студентов: самостоятельное изучение литературы

Краткое содержание материала:

Методы обучение нейронных сетей

Очевидно, что функционирование нейронной сети, т.е. действия, которые она способна выполнять, зависит от величин синаптических связей. Поэтому, задавшись структурой нейронной сети, отвечающей определенной задаче, разработчик должен найти оптимальные значения для всех весовых коэффициентов w. Этот этап называется обучением нейронной сети, и от того, насколько качественно он будет выполнен, зависит способность сети решать во время эксплуатации, поставленные перед ней проблемы. Важнейшими параметрами обучения являются: качество подбора весовых коэффициентов и время, которое необходимо затратить на обучение. Как правило, два этих параметра связаны между собой обратной зависимостью и их приходится выбирать на основе компромисса. В настоящее время все алгоритмы обучения нейронных сетей можно разделить на два больших класса: с учителем и без учителя.

Обучение с учителем. Нейронной сети предъявляются значения как входных, так и выходных параметров, и она по некоторому внутреннему алгоритму подстраивает веса своих синаптических связей. Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются представительской или обучающей выборкой. Обычно нейронная сеть обучается на некотором числе таких выборок. Предъявляется входной вектор, вычисляется выход нейронной сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в нейронную сеть, и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Обучение без учителя. Нейронной сети предъявляются только входные сигналы, а выходы сети формируются самостоятельно с учетом только входных и производных от них сигналов. Несмотря на многочисленные прикладные достижения, обучение с учителем критиковалось за свою биологическую неправдоподобность. Обучение без учителя является более правдоподобной моделью обучения в биологической системе. Развитая Кохоненом и многими другими, она не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса нейронной сети так, чтобы получались согласованные выходные векторы, т.е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Процесс обучения, следовательно, выделяет статистические свойства обучающего множества и группирует сходные векторы в классы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, но до обучения невозможно предсказать, какой выход будет производиться данным классом входных векторов. Следовательно, выходы подобной сети должны трансформироваться в некоторую понятную форму, обусловленную процессом обучения. Это не является серьезной проблемой. Обычно не сложно идентифицировать связь между входом и выходом, установленную сетью.

Архитектура нейронных сетей

Однослойные сети

Реальная нейронная сеть может содержать один или большее количество слоев и соответственно характеризоваться как однослойная или как многослойная. Развернутая схема сети из одного слоя с *R* входными элементами и *S* нейронами показана на рис. 7.1. В этой сети каждый элемент вектора входа соединен со всеми входами нейрона и это

соединение задается матрицей весов **W**; при этом каждый *i*-й нейрон включает суммирующий элемент, который формирует скалярный выход $n(i)$. Совокупность скалярных функций $n(i)$ объединяется в *S*-элементный вектор входа **n** функции активации слоя. Выходы слоя нейронов формируют вектор-столбец **a**, и, таким образом, описание слоя нейронов имеет вид:

$$\mathbf{a=f(W*p+b)}$$

Количество входов *R* в слое может не совпадать с количеством нейронов *S*. В каждом слое, как правило, используется одна и та же функция активации. Однако можно создавать составные слои нейронов с использованием различных функций активации, соединяя сети, параллельно. Обе сети будут иметь те же самые входы, и каждая сеть будет генерировать определенную часть выходов. Элементы вектора входа передаются в сеть через матрицу весов **W**, имеющую вид:

$$W=\begin{pmatrix} w_{11} & w_{12} & \dots & w_{1R} \\ w_{21} & w_{22} & \dots & w_{2R} \\ \dots & \dots & \dots & \dots \\ w_{S1} & w_{S2} & \dots & w_{SR} \end{pmatrix}.$$

Индексы строк матрицы **W** указывают адресатов (пункты назначения) весов нейронов, а индексы столбцов – какой источник является входом для этого веса. Таким образом, элемент матрицы весов w_{12} = **W**(l, 2) определяет коэффициент, на который умножается второй элемент входа при передаче его на первый нейрон.

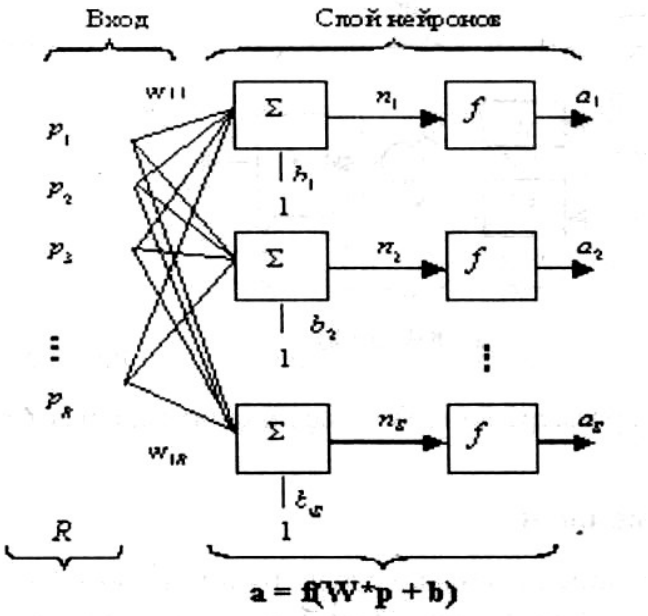


Рис. 7.1. Развернутая схема однослойной сети

Для однослойной сети с *S* нейронами укрупненная структурная схема показана на рис. 7.2.

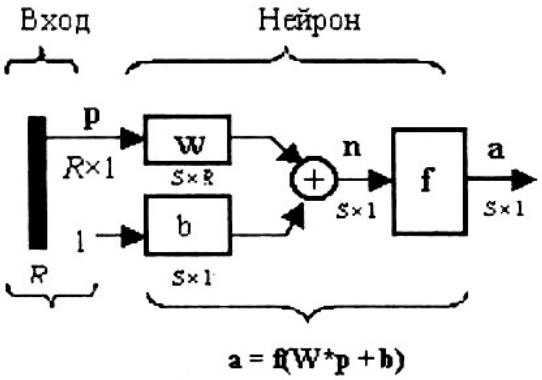


Рис. 7.2. Структурная схема однослойной сети

Здесь **p** – вектор входа размера Rx1**,** **W** – весовая матрица размера SxR, **a, b, n** – векторы размера Sx1.

Многослойные сети

Рассмотрим сети, имеющие несколько слоев. Будем называть весовые матрицы, соединенные с входами, весами входа слоя, а весовые матрицы для сигналов, исходящие из слоя, назовем весами выхода слоя. Далее, будем использовать верхние индексы, чтобы указать источник и адресат для различных весов и других элементов нейронной сети. Чтобы пояснить это, рассмотрим сначала только один, первый слой многослойной сети (рис. 7.3).

Обозначим весовую матрицу, связанную с входами, через $\mathbf{IW}^{11}$**,** верхние индексы которой указывают, что источником входов является первый слой (второй индекс) и адресатом является также первый слой (первый индекс). Элементы этого слоя, такие, как смещение $\mathbf{b}^1$ вход функции активации $\mathbf{n}^1$ и выход слоя $\mathbf{a}^1$, имеют верхний индекс 1, чтобы обозначить, что они связаны с первым слоем.

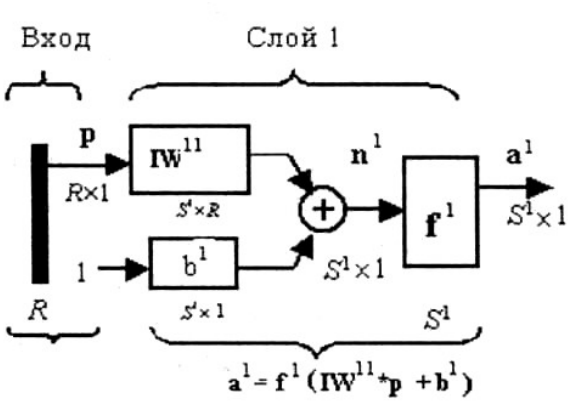


Рис. 7.3. Слой многослойной сети

В дальнейшем для матриц весов входа и выхода слоя будут использованы обозначения **IW**(Input Weight) и **LW**(Layer Weight) соответственно. Когда сеть имеет несколько слоев, то каждый слой имеет свою матрицу весов **W**, вектор смещения **b** и вектор выхода **a**. Чтобы различать весовые матрицы, векторы выхода и т. д. для каждого из этих слоев, введем номер слоя как верхний индекс для представляющей интерес переменной. Сеть, показанная ниже, имеет R входов, S^1 нейронов в первом слое, S^2 нейронов во втором слое и т. д. Для общности будем считать, что различные слои имеют различное число нейронов. На смещения для каждого нейрона подан постоянный входной сигнал 1. Заметим, что выходы каждого промежуточного слоя служат входами для

следующего слоя. Таким образом, слой 2 может быть рассмотрен как один слой сети с S^1 входами, S^2 нейронами и S^1 x S^2 матрицей весов $\mathbf{W_2}$. Вход к слою 2 есть 1, а выход – 2. Теперь, когда обозначены все векторы и матрицы слоя 2, можно трактовать его как самостоятельную однослойную сеть. Такой подход может быть использован к любому слою сети.

Слои многослойной сети имеют различные назначения. Слой, который образует выход сети, называется слоем выхода. Все другие слои называются скрытыми слоями. Трехслойная сеть, имеет выходной слой (слой 3) и 2 скрытых слоя (слой 1 и слой 2). Трехслойная сеть может быть представлена в виде укрупненной структурной схемы (рис. 7.4).

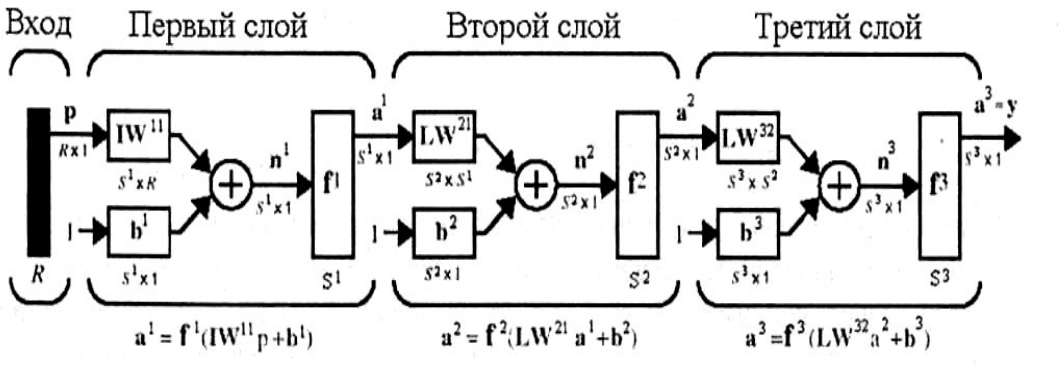


Рис. 7.4. Структурная схема трехслойной сети

Заметим, что выход третьего слоя $\mathbf{a^3}$ обозначен через **y**. Эта сделано для того, чтобы подчеркнуть, что выход последнего слоя является выходом сети. Многослойные сети обладают весьма мощными возможностями. Например, двухслойная сеть, в которой первый слой содержит сигмоидальную, а второй слой линейную функцию активации, может быть обучена аппроксимировать с произвольной точностью любую функцию с конечным числом точек разрыва.

Сети с прямой передачей сигнала

Однослойная сеть с S нейронами с функциями активации logsig, имеющая R входов, показана на рис. 7.5. Эта сеть, не имеющая обратных связей, называется сетью с прямой передачей сигнала. Такие сети часто имеют один или более скрытых слоев нейронов с сигмоидальными функциями активации, в то время как выходной слой содержит нейроны с линейными функциями активации. Сети с такой архитектурой могут воспроизводить весьма сложные нелинейные зависимости между входом и выходом сети.

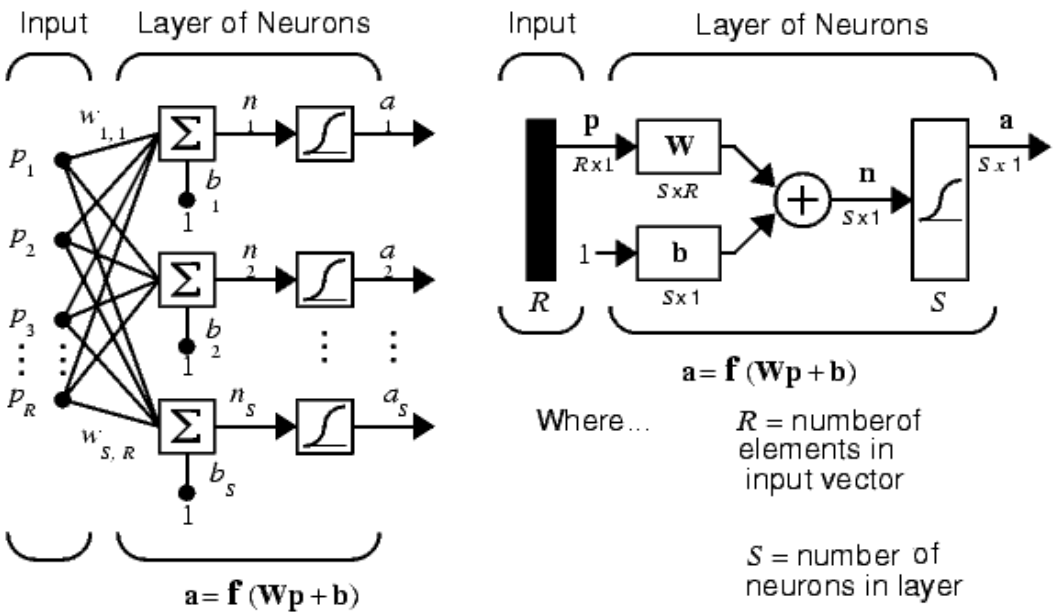


Рис. 7.5. Однослойная сеть с прямой передачей сигнала

Для пояснения обозначений в многослойных нейронных сетях внимательно изучите двухслойную сеть, показанную на рис. 7.6. Эта сеть может быть использована для аппроксимации функций.

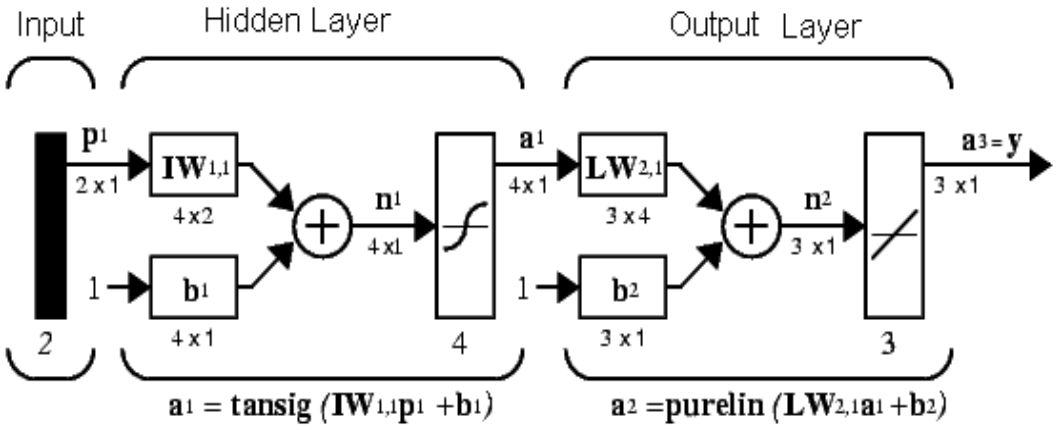


Рис. 7.6. Двухслойная сеть с прямой передачей сигнала

Она может достаточно точно воспроизвести любую функцию с конечным числом точек разрыва, если задать достаточное число нейронов скрытого слоя.

Итоговый продукт самостоятельной работы: конспект
Средства и технологии оценки: собеседование
Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-3	1-3	1-2

Тема самостоятельного изучения № 3.
Программная эмуляция нейрокомпьютеров. Представление нейрона в системе MatLab.
Вид деятельности студентов: самостоятельное изучение литературы
Краткое содержание материала:

Представление нейрона в системе MATLAB

Нейрон со скалярным входом

Элементарной ячейкой нейронной сети является нейрон. Структура нейрона с единственным скалярным входом показана на рис. 6.1,*а*. Скалярный входной сигнал *p* умножается на скалярный весовой коэффициент *w*, и результирующий взвешенный вход *w*p* является аргументом функции активации нейрона *f*, которая порождает скалярный выход *a*.

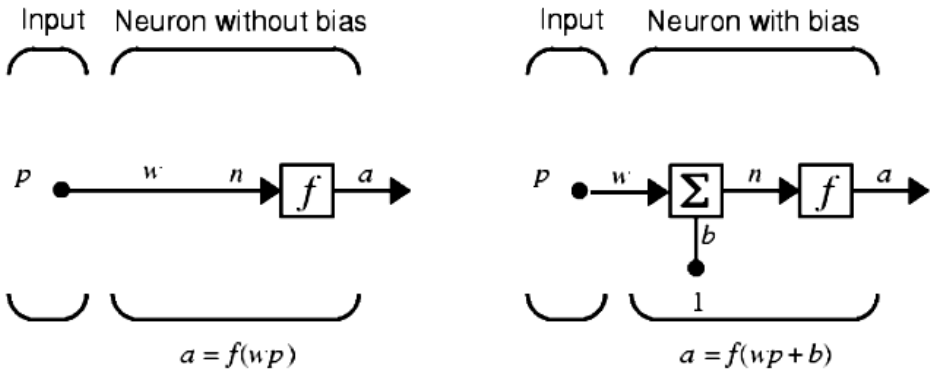


Рис. 6.1. *а*) Нейрон без смещения *б*) Нейрон со смещением

Нейрон, показанный на рис. 6.1, *б*, дополнен скалярным смещением *b*. Смещение суммируется со взвешенным входом *w*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**1). Вход *n* функции активации нейрона по-прежнему остается скалярным и равным сумме взвешенного входа и смещения *b*. Эта сумма (*w*p* + *b**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:

$$a = f(w*p + b*1)$$

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

$$n = \begin{bmatrix} w & b \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = w*p + b*1$$

Нейрон с векторным входом

Нейрон с одним вектором входа **p** с R элементами $p_1, p_2,\ldots, p_R$ показан на рис. 6.2. Здесь каждый элемент входа умножается на веса $w_{11}, w_{12},\ldots, w_{1R}$ соответственно, и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора-строки **W** на вектор-столбец входа **p**. Нейрон имеет смещение b, которое суммируется со взвешенной суммой входов. Результирующая сумма

$$n = w_{11}p_1 + w_{12}\,p_2 + \ldots + w_{1R}\,p_R + b.$$

служит аргументом функции активации f. В нотации языка MATLAB это выражение записывается так: n = **W*p** + b. Структура нейрона, показанная выше, является развернутой. При рассмотрении сетей, состоящих из большого числа нейронов, обычно используется укрупненная структурная схема нейрона (рис. 6.3). Вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа R.

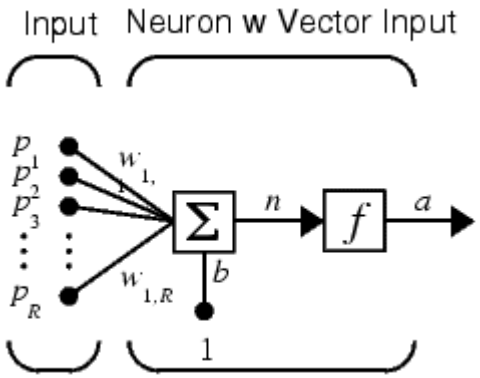


Рис. 6.2. Нейрон с векторным входом

Размер вектора входа **p** указывается ниже символа **p** и равен Rx1. Вектор входа умножается на вектор-строку **W** длины R. Как и прежде, константа 1 рассматривается как вход, который умножается на скалярное смещение b. Входом n функции активации нейрона служит сумма смещения b и произведение **W*p**. Эта сумма преобразуется функцией активации f, на выходе которой получаем выход нейрона a, который в данном случае является скалярной величиной. Структурная схема, приведенная на рис. 6.3, называется *слоем сети*. Слой характеризуется *матрицей весов* **W**, смещением b, операциями умножения **W*p**, суммирования и функцией активации f.

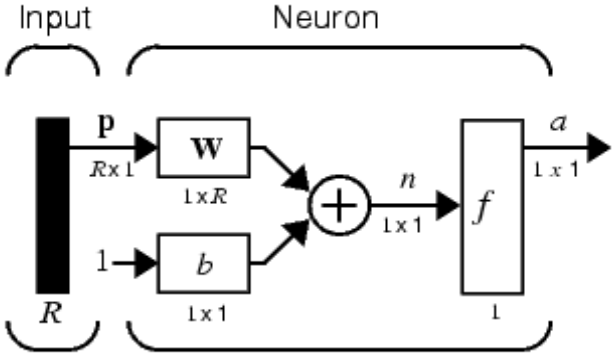


Рис. 6.3. Укрупненная схема нейрона с векторным входом

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. 6.3). Эта система обозначений поясняет строение сети и связанную с ней матричную математику.

Функции активации нейронов

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:

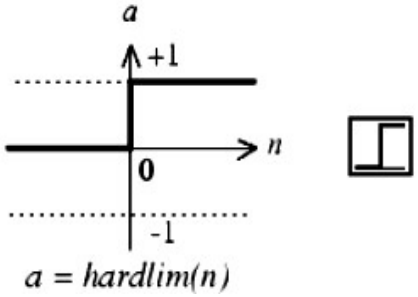


Рис. 6.4. Единичная функция активации

Эта функция описывается соотношением a = hardlim(n) = 1(n) и показана на рис. 6.4. 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+:');
```

Линейная функция активации purelin:

Эта функция описывается соотношением a = purelin(n) = n и показана на рис. 6.5.

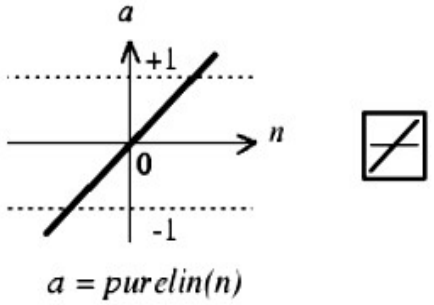


Рис. 6.5. Линейная функция активации

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+:');
```

Логистическая функция активации logsig:

Эта функция описывается соотношением a = logsig(n) = 1/(1 + exp(-n)) и показана на рис. 6.6.

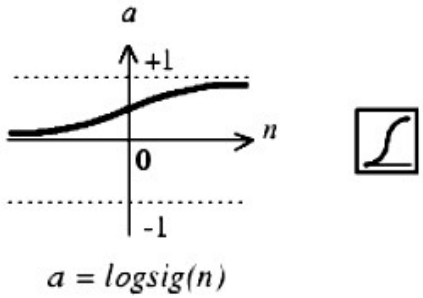


Рис. 6.6. Логистическая функция активации

Данная функция принадлежит к классу сигмоидальных функций, и ее аргумент может принимать любое значение в диапазоне от –∞ до +∞, 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+:');
```

Для построения графика функции одной переменной в системе MATLAB используется оператор plot. При этом графики строятся в отдельных масштабируемых и перемещаемых окнах. Например, для построения графика функции sin *x* достаточно вначале задать диапазон и шаг изменения аргумента, а затем использовать оператор plot:

```
x=-5:0.1:5;
plot(x,sin(x))
```

Оператор plot является мощным инструментом для построения графиков функций одной переменной. Он позволяет строить графики сразу нескольких функций и имеет различные формы, синтаксис которых можно узнать, воспользовавшись командой help plot.

Итоговый продукт самостоятельной работы: конспект
Средства и технологии оценки: собеседование
Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-3	1-3	1-2

Тема самостоятельного изучения № 4.
Программное средство Neural Network Toolbox системы MatLab.
Вид деятельности студентов: самостоятельное изучение литературы
Краткое содержание материала:

NNTool – инструментальное средство с графическим интерфейсом, позволяющее начинающему пользователю пакета Neural Network Toolbox выполнять создание, обучение, моделирование, экспорт и импорт нейронных сетей и данных, не обращаясь к командному окну MATLAB. Вызов NNTool возможен либо с помощью команды nntool из командной строки MATLAB, либо из окна запуска приложений Launch Pad с помощью опции NNTool из раздела Neural Network Toolbox. После вызова на экране дисплея появляется окно Network/Data Manager (Управление сетью/Данными) (рис. 1).
Элементы окна:
Help – кнопка вызова окна подсказки Network/ Data Manager Help;
New Data – кнопка вызова окна формирования данных Create New Data (рис. 2);

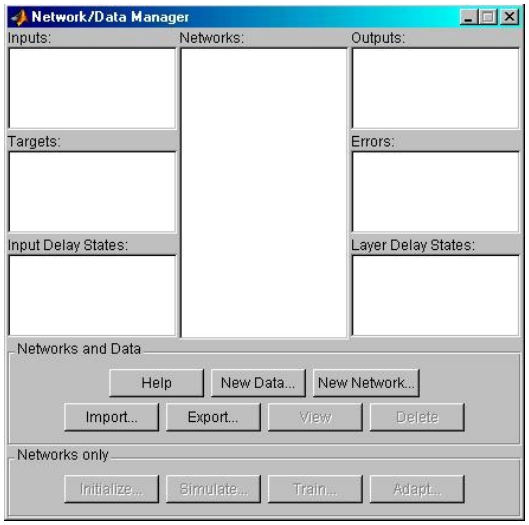


Рис. 1

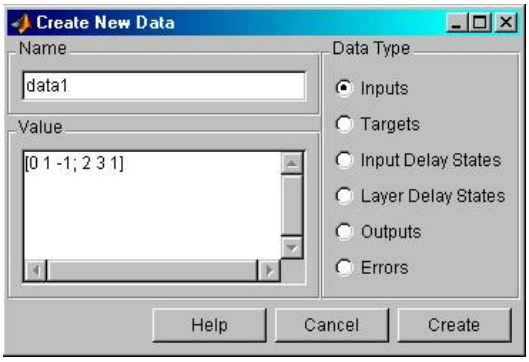


Рис.2

New Network…– кнопка вызова окна создания нейронной сети Create New Network (рис. 3);
Import… – кнопка вызова окна для импорта или загрузки данных Import or Load to Network/Data Manager (рис. 4);
Export… – кнопка вызова окна для экспорта или записи данных в файл Export or Save from Network/Data Manager;
Кнопки View, Delete становятся активными после создания и активизации данных, относящихся к входам, целям, выходам или ошибкам сети.

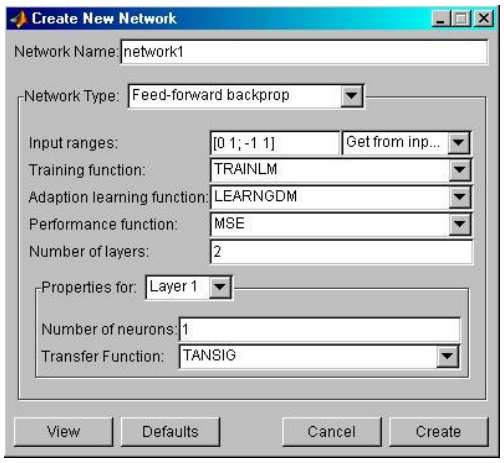


Рис. 3

Чтобы создать нейронную сеть, необходимо:

5) сформировать последовательности входов и целей (кнопка New Data) либо загрузить их из рабочей области системы MATLAB;
6) создать новую нейронную сеть (кнопка New Network) либо загрузить ее из рабочей области или из файла (кнопка Import);
7) выбрать тип нейронной сети и нажать кнопку Train, чтобы открыть окно для задания параметров процедуры обучения;
8) открыть окно Network для просмотра, инициализации, моделирования и адаптации сети.

Network Name – стандартное имя сети, присваиваемое GUI-интерфейсом, порядковый номер имени изменяется автоматически.

Network Type – тип сетей, доступных для работы в NNTool.

Inputs ranges – допустимые границы входов, которые задаются пользователем либо определяются автоматически из списка Get from Inp…

Training function – список обучающих функций.

Adaptation learning function – список функций настроек для режима адаптации.

Performance function – список функций оценки качества обучения.

Number of layers – количество слоев сети.

Properties for – свойства для слоев 1 и 2.

Numbers of neurons – количество нейронов в слое.

Transfer function – функция активации слоя.

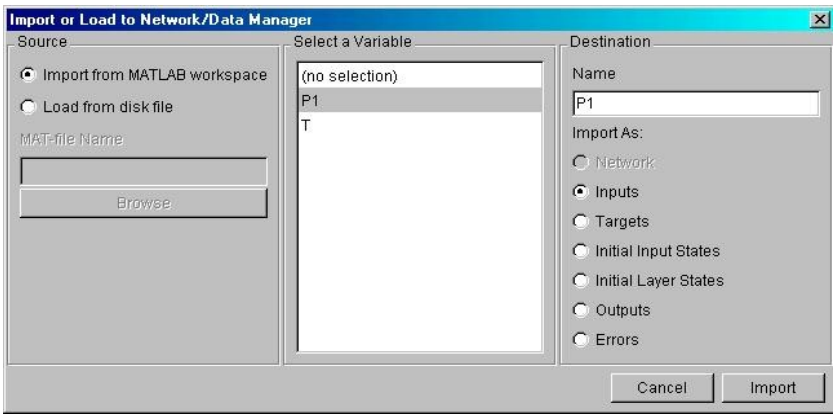


Рис. 4

Окно Import or Load to Network/ Data Manager включает три поля:

Source – поле для выбора источника данных: рабочая область MATLAB (Import from MATLAB workspace) либо файл (Load from disk file).

При выборе первого варианта поле Select a variable содержит все переменные рабочей области. При выборе второго варианта активизируется поле MAT – file Name и кнопка Browse.

Destination – поле, в котором следует указать тип импортируемых данных путем выбора одной из радиокнопок.

Окно Export or Save from Network/Data Manager (рис. 5) позволяет передать данные из рабочей области NNTool в рабочую область MATLAB или сохранить их в виде файла на диске. Если в качестве сохраняемой переменной выбрана переменная, принадлежащая классу network object, и она экспортирована в рабочую область MATLAB, можно построить модель нейронной сети в системе Simulink.

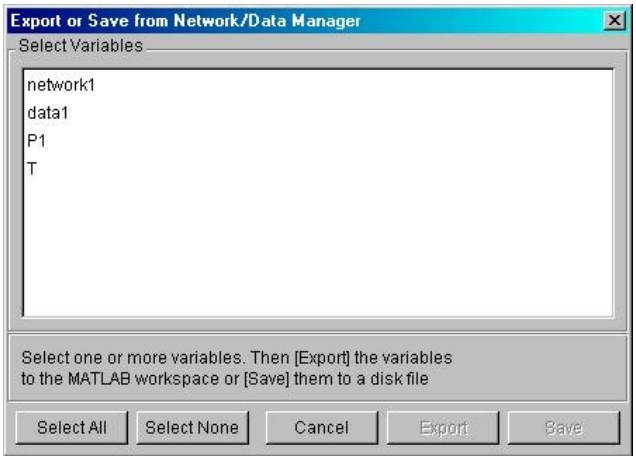


Рис. 5

После создания и активизации нейронной сети в окне Network/Data Manager становятся активными кнопки View, Delete, Initialize, Simulate, Train, Adapt. При выделении в области Networks созданной сети открывается диалоговая панель Network: <имя_сети> (рис. 6).

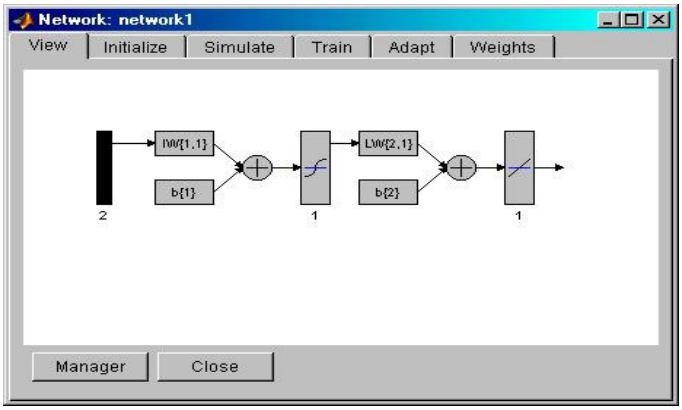


Рис. 6

Панель имеет шесть закладок:
View – просмотр структуры сети.
Initialize – задание начальных весов и смещений.
Simulate – моделирование сети.
Train – обучение сети.
Adapt – адаптация и настройка параметров в сети.
Weights – просмотр установленных весов и смещений.

Итоговый продукт самостоятельной работы: конспект
Средства и технологии оценки: собеседование
Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-3	1-3	1-2

Тема самостоятельного изучения № 6.
Самоорганизующиеся и рекуррентные сети. Гибридные нейронные сети.
Вид деятельности студентов: самостоятельное изучение литературы

Самоорганизующиеся сети

Сети Кохонена

В процессе анализа больших информационных массивов данных неизменно возникают задачи, связанные с исследованием топологической структуры данных, их объединением в группы (кластеры), распределением по классам и т. п. Это могут быть экономические, финансовые, научно-технические, медицинские и другие приложения, где требуется решение таких практических задач, как сжатие данных, их хранение и поиск, определение характеристик объекта по ограниченному набору признаков. Такие задачи могут быть успешно решены с применением специального класса самоорганизующихся нейронных сетей. Нейронные сети Кохонена или *самоорганизующиеся карты Кохонена* (*Kohonen's Self-Organizing Maps*) предназначены для решения задач автоматической классификации, когда обучающая последовательность образов отсутствует. Соответственно отсутствует и фиксация ошибки, на минимизации которой основаны алгоритмы обучения, например, алгоритм обратного распространения ошибки (*Backpropagation*).

Сеть Кохонена – это двухслойная нейронная сеть, содержащая *входной слой* (слой входных нейронов) и *слой Кохонена* (слой активных нейронов). Слой Кохонена может быть: одномерным, двумерным или трехмерным. В первом случае активные нейроны расположены в цепочку. Во втором случае они образуют двухмерную сетку (обычно в форме квадрата или прямоугольника), а в третьем случае они образуют трехмерную конструкцию. В силу отсутствия обучающей последовательности образов, для каждого из которых известна от учителя принадлежность к тому или иному классу, определение весов нейронов слоя Кохонена основано на использовании алгоритмов классической классификации (кластеризации или самообучения). Свойство самоорганизации является одним из наиболее привлекательных свойств нейронных сетей. Нейроны самоорганизующейся сети могут быть обучены выявлению групп (кластеров) векторов входа, обладающих некоторыми общими свойствами. При изучении самоорганизующихся нейронных сетей, или *сетей Кохонена,* существенно различать сети с неупорядоченными нейронами, которые часто называют *слоями Кохонена* и сети с упорядочением нейронов, которые часто называют *картами Кохонена.* Последние отражают структуру данных таким образом, что близким кластерам данных на карте соответствуют близко расположенные нейроны.

Слой Кохонена

Рассмотрим самоорганизующуюся нейронную сеть с единственным слоем, задача которой заключается в том, чтобы правильно сгруппировать (кластеризировать) поступающие на нее векторы входа.

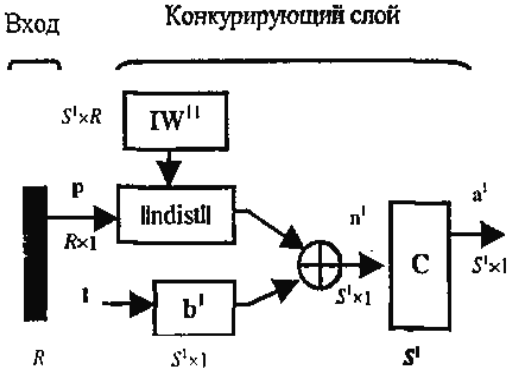


Рис. 12.1. Архитектура слоя Кохонена

Нетрудно убедиться, что это слой конкурирующего типа, поскольку в нем применена конкурирующая функция активации. Кроме того, архитектура этого слоя очень напоминает архитектуру скрытого слоя радиальной базисной сети. Здесь использован блок ndist для вычисления отрицательного евклидова расстояния между вектором входа **p** и строками матрицы весов $\mathbf{IW}^1$. Вход функции активации $\mathbf{n}^1$ – это результат суммирования вычисленного расстояния с вектором смещения ***b***. Если все смещения нулевые, максимальное значение $\mathbf{n}^1$ не может превышать 0. Нулевое значение $\mathbf{n}^1$ возможно только, когда вектор входа **p** оказывается равным вектору веса одного из нейронов. Если смещения отличны от 0, то возможны и положительные значения для элементов вектора $\mathbf{n}^1$.

Конкурирующая функция активации анализирует значения элементов вектора $\mathbf{n}^1$ и формирует выходы нейронов, равные 0 для всех нейронов, кроме одного нейрона-победителя, имеющего на входе максимальное значение. Таким образом, вектор выхода слоя $\mathbf{a}^1$ имеет единственный элемент, равный 1, который соответствует нейрону-победителю, а остальные равны 0. Такая активационная характеристика может быть описана следующим образом:

$$\mathbf{a}_i^1 = \begin{cases} 1, & i = i^*,\ i^* = \arg(\max \mathbf{n}_i^1) \\ 0, & i \neq i^* \end{cases}$$

Заметим, что эта активационная характеристика устанавливается не на отдельный нейрон, а на слой. Поэтому такая активационная характеристика и получила название *конкурирующей.* Номер активного нейрона *i* определяет ту группу (кластер), к которой наиболее близок входной вектор. Для формирования слоя Кохонена предназначена М-функция newc.

Покажем, как она работает, на простом примере. Предположим, что задан массив из четырех двухэлементных векторов, которые надо разделить на 2 класса:

```
p =  [.1  .8  .1  .9;   .2  .9  .1  .8]
P =
0.1000 0.8000 0.1000 0.9000
0.2000 0.9000 0.1000 0.8000
```

В этом примере нетрудно видеть, что 2 вектора расположены вблизи точки (0,0) и 2 вектора – вблизи точки (1,1). Сформируем слой Кохонена с двумя нейронами для анализа двухэлементных векторов входа с диапазоном значений от 0 до 1:

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

Первый аргумент указывает диапазон входных значений, второй определяет количество нейронов в слое. Начальные значения элементов матрицы весов задаются как среднее максимального и минимального значений, т. е. в центре интервала входных значений; это реализуется по умолчанию с помощью М-функции midpoint при создании сети. Убедимся, что это действительно так:

```
wts = net.iw{l,l}
wts =
0.5000 0.5000
0.5000 0.5000
```

Определим характеристики слоя Кохонсна

```
net.layers{1}
ans =
dimensions:    2
distanceFcn:   'dist'
distances:     [2x2 double}
```

initFcn: 'initwb'
netlnputFcn: 'netsum'
positions: [0 1]
size: 2
topologyFcn: 'hextop[1]
transferFcn: 'compet'
userdata: [lxl struct]

Из этого описания следует, что сеть использует функцию евклидова расстояния dist, функцию инициализации initwb, функцию обработки входов netsum, функцию активации compet и функцию описания топологии hextop.

Характеристики смещений следующие:

net.biases{1}
ans =
initFcn: 'initcon'
learn: 1
learnFcn: 'learncon'
learnParam: [lxl struct]
size: 2
userdata: [lxl struct]

Смещения задаются функцией initcon и для инициализированной сети равны

net.Ь{1}
ans =
5.4366 5.4366

Функцией настройки смещений является функция learncon, обеспечивающая настройку с учетом параметра активности нейронов. Теперь, когда сформирована самоорганизующаяся нейронная сеть, требуется обучить се решению задачи кластеризации данных. Напомним, что каждый нейрон блока compet конкурирует за право ответить на вектор входа р. Если все смещения равны 0, то нейрон с вектором веса, самым близким к вектору входа р, выигрывает конкуренцию и возвращает на выходе значение 1; все другие нейроны возвращают значение 0.

Правила обучения и настройки смещений слоя Кохонена

Правило обучения слоя Кохонена, называемое также правилом Кохонена, заключается в том, чтобы настроить нужным образом элементы матрицы весов. Предположим, что нейрон i победил при подаче входа $\mathbf{p}(q)$ на шаге самообучения q, тогда строка i матрицы весов корректируется в соответствии с правилом Кохонена следующим образом:

$$\mathbf{IW}_{11}^{i}(q) = \mathbf{IW}_{11}^{i}(q-1) + \alpha(\mathbf{p}(q) - \mathbf{IW}_{11}^{i}(q-1)).$$

Правило Кохонена представляет собой рекуррентное соотношение, которое обеспечивает коррекцию строки i матрицы весов добавлением взвешенной разности вектора входа и значения строки на предыдущем шаге. Таким образом, вектор веса, наиболее близкий к вектору входа, модифицируется так, чтобы расстояние между ними стало еще меньше. Результат такого обучения будет заключаться в том, что победивший нейрон, вероятно, выиграет конкуренцию и в том случае, когда будет представлен новый входной вектор, близкий к предыдущему, и его победа менее вероятна, когда будет представлен вектор, существенно отличающийся от предыдущего. Когда на вход сети поступает все большее и большее число векторов, нейрон, являющийся ближайшим, снова корректирует свой весовой вектор. В конечном счете, если в слое имеется достаточное количество нейронов, то каждая группа близких векторов окажется связанной с одним из нейронов слоя. В этом и заключается свойство самоорганизации слоя Кохонена.

Правило настройки смещений. Одно из ограничений всякого конкурирующего слоя состоит в том, что некоторые нейроны оказываются незадействованными. Это

проявляется в том, что нейроны, имеющие начальные весовые векторы, значительно удаленные от векторов входа, никогда не выигрывают конкуренции, независимо от того как долго продолжается обучение. В результате оказывается, что такие векторы не используются при обучении и соответствующие нейроны никогда не оказываются победителями. Такие нейроны-неудачники называют "мертвыми" нейронами, поскольку они не выполняют никакой полезной функции. Чтобы исключить такую ситуацию и сделать нейроны чувствительными к поступающим на вход векторам, используются смещения, которые позволяют нейрону стать конкурентным с нейронами-победителями. Этому способствует положительное смещение, которое добавляется к отрицательному расстоянию удаленного нейрона. Соответствующее правило настройки, учитывающее нечувствительность мертвых нейронов, реализовано в виде М-функции learncon и заключается в следующем. В начале процедуры настройки всем нейронам конкурирующего слоя присваивается одинаковый параметр активности

$$c_0 = \frac{1}{N},$$

где N – количество нейронов конкурирующего слоя, равное числу кластеров. В процессе настройки М-функция learncon корректирует этот параметр таким образом, чтобы его значения для активных нейронов становились больше, а для неактивных нейронов меньше. Соответствующая формула для вектора приращении параметров активности выглядит следующим образом:

$$\Delta\mathbf{c} = lr*(\mathbf{a}_i^1 - \mathbf{c}),$$

где lr – параметр скорости настройки; $\mathbf{a}_i^1$ – вектор, элемент $i*$ которого равен 1, а остальные 0. Нетрудно убедиться, что для всех нейронов, кроме нейрона-победителя, приращения отрицательны. Поскольку параметры активности связаны со смещениями соотношением

$\mathbf{b} = \exp(1)/\mathbf{c}$,

то из этого следует, что смещение для нейрона-победителя уменьшится, а смещения для остальных нейронов немного увеличатся. М-функция learncon использует следующую формулу для расчета приращений вектора смещений

$\Delta\mathbf{b} = \exp(1 - \log(\mathbf{c})) - \mathbf{b}$.

Параметр скорости настройки lr по умолчанию равен 0.001, и его величина обычно на порядок меньше соответствующего значения для М-функции learnk. Увеличение смещений для неактивных нейронов позволяет расширить диапазон покрытия входных значений, и неактивный нейрон начинает формировать кластер. В конечном счете, он может начать притягивать новые входные векторы. Это дает два преимущества. Если нейрон не выигрывает конкуренции, потому что его вектор весов существенно отличается от векторов, поступающих на вход сети, то его смещение по мере обучения становится достаточно большим, и он становится конкурентоспособным. Когда это происходит, его вектор весов начинает приближаться к некоторой группе векторов входа. Как только нейрон начинает побеждать, его смещение начинает уменьшаться. Таким образом, задача активизации "мертвых" нейронов оказывается решенной. Второе преимущество, связанное с настройкой смещений, состоит в том, что они позволяют выровнять значения параметра активности и обеспечить притяжение приблизительно одинакового количества векторов входа. Таким образом, если один из кластеров притягивает большее число векторов входа, чем другой, то более заполненная область притянет дополнительное количество нейронов и будет поделена.

Карта Кохонена

Самоорганизующаяся сеть в виде карты Кохонена предназначена для решения задач кластеризации входных векторов. В отличие от слоя Кохонена карта Кохонена

поддерживает такое топологическое свойство, когда близким кластерам входных векторов соответствуют близко расположенные нейроны.
Первоначальная топология размещения нейронов в слое Кохонена задается М-функциями gridtop, hextop или randtop, что соответствует размещению нейронов в узлах либо прямоугольной, либо гексагональной сетки, либо в узлах сетки со случайной топологией.

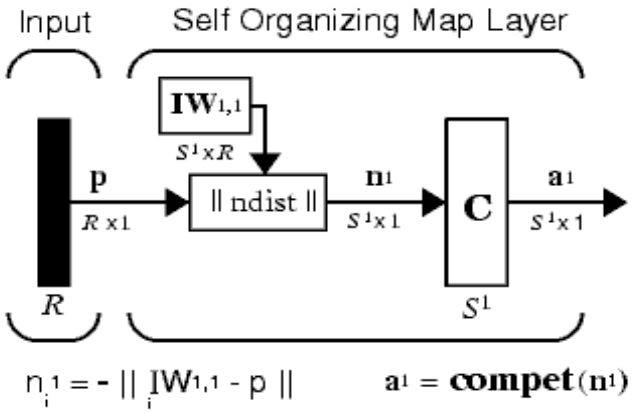


Рис. 12.2. Архитектура карты Кохонена

Расстояния между нейронами вычисляются с помощью специальных функций вычисления расстояний dist, boxdist, linkdist и mandist. Карта Кохонена для определения нейрона-победителя использует ту же процедуру, какая применяется и в слое Кохонена. Однако на карте Кохонена одновременно изменяются весовые коэффициенты соседних нейронов в соответствии со следующим соотношением, именуемым правилом Кохонена:
$w_i(q) = (1 - \alpha)_i\, w(q - 1) + \alpha p(q)$.
В этом случае окрестность нейрона-победителя включает все нейроны, которые находятся в пределах некоторого радиуса d:
$N_i(d) = \{j, d_{ij} < d\}$.

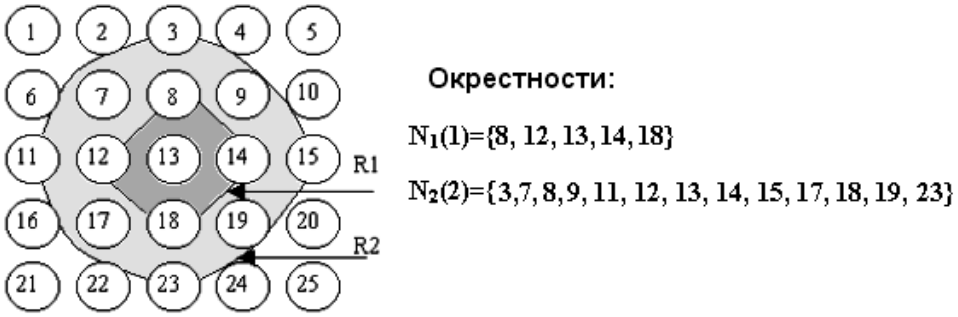


Рис. 12.3. Окружности радиусов 1 и 2 нейрона-победителя с номером 13.

Топология карты расположения нейронов не обязательно должна быть двумерной. Это могут быть и одномерные и трехмерные карты, и даже карты больших размерностей. В случае одномерной карты Кохонена, когда нейроны расположены вдоль линии, каждый нейрон будет иметь только двух соседей в пределах радиуса 1 или единственного соседа, если нейрон расположен на конце линии. Расстояния между нейронами можно определять различными способами, используя прямоугольные или гексагональные сетки, однако это никак не влияет на характеристики сети, связанные с классификацией входных векторов. Можно задать различные топологии для карты расположения нейронов, используя М-функции gridtop, hextop, randtop. Рассмотрим простейшую прямоугольную сетку размера

2x3 для размещения шести нейронов, которая может быть сформирована с помощью функции gridtop:

```
pos = gridtop{2,3)
pos =
0 1 0 1 0 1
0 0 1 1 2 2
plotsom(pos) %
```

Здесь нейрон 1 расположен в точке с координатами (0,0), нейрон 2 – в точке (1,0), нейрон 3 – в точке (0,1) и т. д. Заметим, что, если применить команду gridtop, переставив аргументы местами, получим иное размещение нейронов:

```
pos = gridtop(3,2)
pos =
0 1 2 0 1 2
0 0 0 1 1 1
```

Гексагональную сетку можно сформировать с помощью функции hextop:

```
pos = hextop(2,3)
pos =
0  1.0000  0.5000  1.5000       0   1.0000
0       0  0.8660  0.8660   1.7321   1.7321
plotsom(pos)
```

Заметим, что М-функиия hextop используется по умолчанию при создании карт Кохонена при применении функции newsom. Сетка со случайным расположением узлов может быть создана с помощью функции randtop.

Рекуррентные сети

Сети Элмана

В этой главе рассматриваются 2 типа рекуррентных нейронных сетей, представляющих наибольший интерес для пользователей, – это класс сетей Элмана (Elman) и класс сетей Хопфилда (Hopfield). Характерной особенностью архитектуры рекуррентной сети является наличие блоков динамической задержки и обратных связей. Это позволяет таким сетям обрабатывать динамические модели. Обратимся к описанию конкретных типов рекуррентных нейронных сетей.

Сеть Элмана – это сеть, состоящая из двух слоев, в которой скрытый слой охвачен динамической обратной связью. Это позволяет учесть предысторию наблюдаемых процессов и накопить информацию для выработки правильной стратегии управления. Сети Элмана применяются в системах управления движущимися объектами, при построении систем технического зрения и в других приложениях. В качестве функций активации в сети Элмана часто используются: в скрытом, рекуррентном слое – функция гиперболического тангенса tansig, в линейном слое – функция purelin. Такое сочетание функций активации позволяет максимально точно аппроксимировать функции с конечным числом точек разрыва. Единственное требование, предъявляемое к сети, состоит в том, чтобы скрытый слой имел достаточно большое число нейронов, что необходимо для успешной аппроксимации сложных функций.

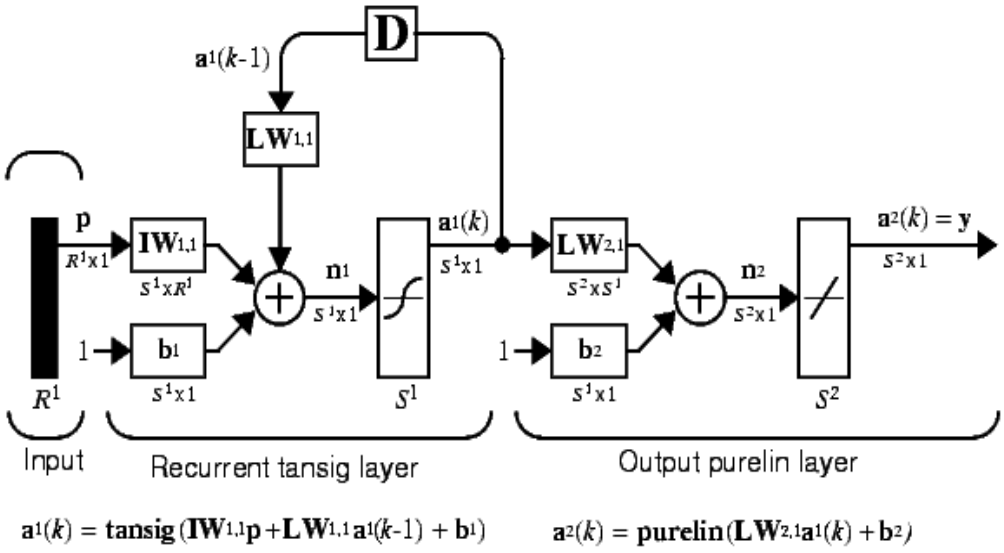


Рис. 13.1. Архитектура сети Элмана

В соответствии со структурной схемой сети Элмана сформируем динамическое описание ее рекуррентного слоя в виде уравнений состояния

$$\begin{cases} \mathbf{n}^1(k) = \mathbf{LW}^{11}\mathbf{a}^1(k-1) + \mathbf{IW}^{11}\mathbf{p} + \mathbf{b}^1, \quad \mathbf{a}^1(0) = \mathbf{a}_0^1; \\ \mathbf{a}^1(k) = \text{tansig}(\mathbf{n}^1(k)). \end{cases}$$

Эта рекуррентная матричная форма уравнений состояния лишний раз подчеркивает название изучаемых нейронных сетей. Второй, линейный слой является безынерционным и описывается соотношениями

$$\begin{cases} n^2(k) = \mathbf{LW}^{21}\mathbf{a}^1(k) + \mathbf{b}^2; \\ \mathbf{a}^2(k) = \text{purelin}(\mathbf{n}^2(k)). \end{cases}$$

Сети Хопфилда

Всякий целевой вектор можно рассматривать как набор характерных признаков некоторого объекта. Если создать рекуррентную сеть, положение равновесия которой совпадало бы с этим целевым вектором, то такую есть можно было бы рассматривать как ассоциативную память. Поступление на вход такой сети некоторого набора признаков в виде начальных условий приводило бы ее в то или иное положение равновесия, что позволяло бы ассоциировать вход с некоторым объектом. Именно такими ассоциативными возможностями и обладают *сети Хопфилда.* Они относятся к классу рекуррентных нейронных сетей, обладающих тем свойством, что за конечное число тактов времени они из произвольного начального состояния приходят в состояние устойчивого равновесия, называемое *аттрактором.* Количество таких аттракторов определяет объем ассоциативной памяти сети Хопфилда.

Спроектировать сеть Хопфилда – это значит создать рекуррентную сеть с множеством точек равновесия, таких, что при задании начальных условий сеть, в конечном счете, приходит в состояние покоя в одной из этих точек. Свойство рекурсии проявляется в том, что выход сети подается обратно на вход. Можно надеяться, что выход сети установится в одной из точек равновесия. Синтезируемая сеть в дополнение к желаемым, может иметь *паразитные точки равновесия.* Однако число таких паразитных

точек должно быть сведено к минимуму за счет конструирования метода синтеза. Более того, область притяжения точек равновесия должна быть максимально большой. Метод синтеза сети Хопфилда основан на построении системы линейных дифференциальных уравнений первого порядка, которая задана в некотором замкнутом гиперкубе пространства состояний и имеет решения в вершинах этого гиперкуба. Такая сеть несколько отличается от классической модели Хопфилда, но она проще для понимания и проектирования.

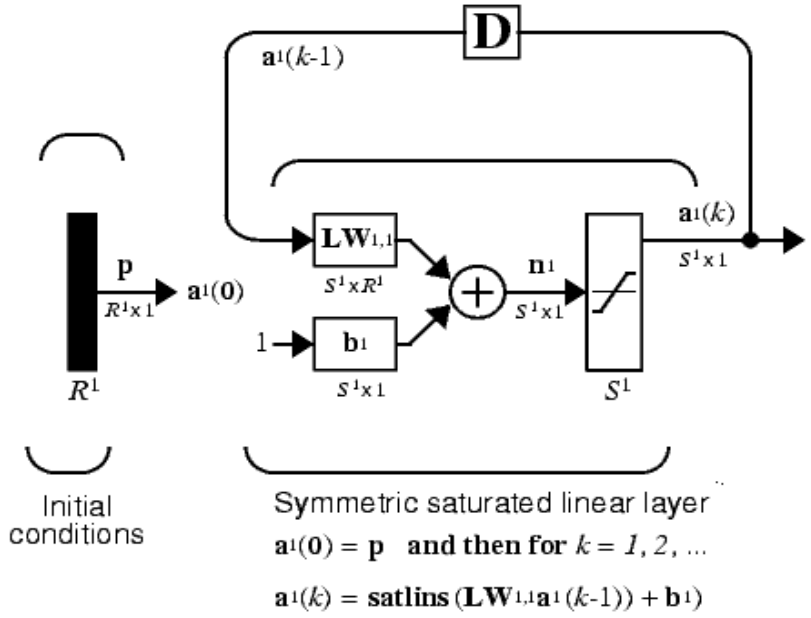


Рис. 13.2. Архитектура сети Хопфилда

Вход *p* устанавливает значения начальных условий. В сети используется линейная функция активации с насыщением satlins, которая описывается следующим образом:

$$a = satlins(n) = \begin{cases} -1, & n < -1; \\ n, & -1 \le n \le 1; \\ 1, & n > 1. \end{cases}$$

Эта сеть может быть промоделирована с одним или большим количеством векторов входа, которые задаются как начальные условия. После того как начальные условия заданы, сеть генерирует выход, который по обратной связи подается на вход. Этот процесс повторяется много раз, пока выход не установится в положение равновесия. Можно надеяться, что каждый вектор выхода, в конечном счете, сойдется к одной из точек равновесия, наиболее близкой к входному сигналу.

Итоговый продукт самостоятельной работы: конспект
Средства и технологии оценки: собеседование
Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-3	1-3	1-2

4. КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Оценка «отлично» выставляется студенту, если глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.

Оценка «неудовлетворительно» выставляется студенту, если неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы.

9. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

9.1. Рекомендуемая литература

9.1.1. Основная литература:

3. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов. – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2013. – 400 с.
4. Медведев В.С., Потемкин В.Г. Нейронные сети. MATLAB 6 / Под общ. ред. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2012. – 496 с.

9.1.2. Дополнительная литература:

1. Галушкин А.И. Теория нейронных сетей. – М.: ИПРЖР, 2000. – 416 с.
2. Хайкин С. Нейронные сети. Полный курс. – М.: Изд. дом Вильямс, 2006.
3. Чернышев А.Б., Козлов В.А., Жерносек И.А. Нейрокомпьютерные сети: Лабораторный практикум. – Пятигорск: Изд-во ПГТУ, 2011. – 40 с.

9.1.3. Методическая литература:

4. Методические указания по выполнению лабораторных работ по дисциплине «Теория нейронных сетей».
5. Методические указания по выполнению контрольной работы по дисциплине «Теория нейронных сетей».
6. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Теория нейронных сетей».

9.1.4. Интернет-ресурсы:

3. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
4. http://window.edu.ru – образовательные ресурсы ведущих вузов

9.1.5. Программное обеспечение

2. Neural Network Toolbox системы MatLab.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

для обучающихся по выполнению контрольной работы
по дисциплине «ТЕОРИЯ НЕЙРОННЫХ СЕТЕЙ» для студентов направления
подготовки **09.04.02 Информационные системы и технологии**
направленность (профиль) **Технологии работы с данными и знаниями, анализ информации**

Пятигорск
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 58

1. ФОРМУЛИРОВКА ЗАДАНИЯ И ЕГО ОБЪЕМ 58
2. ОБЩИЕ ТРЕБОВАНИЯ К НАПИСАНИЮ И ОФОРМЛЕНИЮ РАБОТЫ 58
3. ВАРИАНТЫ ЗАДАНИЙ ДЛЯ СТУДЕНТОВ ЗАОЧНОЙ ФОРМЫ ОБУЧЕНИЯ 58
4. ПРИМЕР ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ 59
5. КРИТЕРИИ ОЦЕНИВАНИЯ РАБОТЫ 67
6. ПОРЯДОК ЗАЩИТЫ РАБОТЫ 68
7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 68

ВВЕДЕНИЕ

Методические указания содержат перечень вариантов заданий для контрольных работ, требования к оформлению контрольных работ и пример выполнения задания. Теоретической основой подготовки специалиста являются знания в области информатики, вычислительной систем.

1. ФОРМУЛИРОВКА ЗАДАНИЯ И ЕГО ОБЪЕМ

Контрольная работа включает в себе вопросы по темам, которые нужно изучить самостоятельно и по ним подготовить отчет. Результаты выполнения контрольной работы предоставляются в электронном виде и распечатанном виде. Объем контрольной работы составляет 10-15 печатных листов формата А4.

Варианты заданий выбираются из таблицы по последней цифре зачетной книжки.

2. ОБЩИЕ ТРЕБОВАНИЯ К НАПИСАНИЮ И ОФОРМЛЕНИЮ РАБОТЫ

Контрольная работа выполняется и сдается в электронном виде на CD/CDRW носителе и в распечатанном виде. На конверте необходимо указать название дисциплины, ФИО студента, факультет, номер группы, шифр зачетной книжки, № варианта задания, и список всех созданных в ходе выполнения задания файлов.

Приведенный в конце методических указаний список литературы может использоваться студентами при выполнении контрольной работы.

3. ВАРИАНТЫ ЗАДАНИЙ ДЛЯ СТУДЕНТОВ ЗАОЧНОЙ ФОРМЫ ОБУЧЕНИЯ

Задание 1. Описать принципы построения и обучения нейронной сети:

1. Персептрон
2. Статическая линейная сеть
3. Динамическая линейная сеть
4. Радиальная базисная сеть
5. Слой Кохонена
6. Карта Кохонена
7. Рекуррентная сеть Элмана
8. Сеть Хопфилда
9. Сеть с прямой передачей сигнала
10. Гибридные нейронные сети

Задание 2. Используя средство NNTool MATLAB, создать и обучить нейронную сеть выполнению операции аппроксимации функции, если заданы последовательности входа и цели.

9) $y=x_1^2-x_2^2+x_3$.
10) $y=x_1+x_2+x_3^2$.
11) $y=x_1^2+x_2+x_3^2$.
12) $y=x_1^2-x_2+x_3^2$.
13) $y=x_1^2+x_2-x_3^2$.
14) $y=x_1+x_2-x_3^2$.
15) $y=x_1^2-x_2^2+x_3^2$.
16) $y=x_1^2+x_2^3+x_3^2$.
17) $y=x_1^3+x_2+x_3^2$.
18) $y=x_1^3+x_2^3+x_3^2$

Диапазон изменения параметров: $x_1=[-2;2]$; $x_2=[-1;1]$; $x_3=[0;4]$.

Задание 3. Реализовать в MATLAB моделирование персептронной сетью заданной логической функции.

1. $(A \vee B) \to (A \wedge B)$
2. $\overline{(A \vee B) \vee \overline{A}}$
3. $\overline{(A \vee B) \vee (B \leftrightarrow A)}$
4. $\overline{A \to B} \vee (\bar{A} \wedge B)$
5. $(\overline{A} \leftrightarrow B) \wedge \overline{A \wedge B}$
6. $((A \to B) \wedge B) \to A$
7. $(A \to B) \vee A \to B$
8. $\overline{B} \wedge (A \to B) \to \overline{A}$
9. $(A \to B) \leftrightarrow (\overline{B} \to \overline{A})$
10. $\overline{A} \wedge (A \vee B) \to B$

Для определения функции цели можно воспользоваться таблицей:

A	B	$\bar{A}$	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

4. ПРИМЕР ВЫПОЛНЕНИЯ КОНТРОЛЬНОЙ РАБОТЫ

<u>Задание 1.</u> Описать принципы построения и обучения нейронной сети:
РАДИАЛЬНАЯ БАЗИСНАЯ СЕТЬ

В общем случае под радиальной базисной нейронной сетью (Radial Basis Function Network, сеть RBF) понимается двухслойная сеть без обратных связей, которая содержит скрытый слой радиально симметричных нейронов. Радиальные базисные нейронные сети состоят из большего количества нейронов, чем стандартные сети с прямой передачей сигналов и обучением методом обратного распространения ошибки, но на их создание требуется значительно меньше времени. Эти сети наиболее эффективны, когда доступно большое количество обучающих векторов. На рис. 1. показан радиальный базисный нейрон с R входами. Функция активации для радиального базисного нейрона имеет вид

$$a = radbas(n) = e^{-n^2}$$

Вход функции активации определяется как модуль разности вектора весов $\boldsymbol{w}$ и вектора входа $\boldsymbol{p}$, умноженный на смещение b.

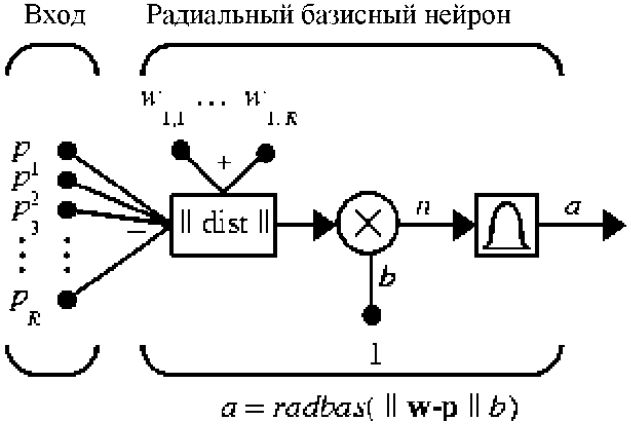


Рис. 1. Радиальный базисный нейрон

График функции активации представлен на рис. 2.

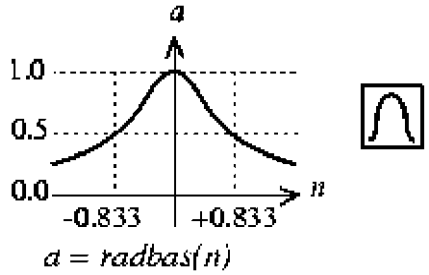


Рис. 2. График функции активации

Вход функции активации определяется как модуль разности вектора весов ***w*** и вектора входа ***p,*** умноженный на смещение *b:*

$$n = \|\mathbf{p} - \boldsymbol{w}\| b.$$

Радиальный базисный нейрон формирует значение *1,* когда вход ***p*** идентичен вектору весов ***w***. Смещение *b* позволяет корректировать чувствительность нейрона radbas. Например, если нейрон имел смещение 0.1, то его выходом будет 0.5 для любого вектора входа p и вектора веса ***w*** при расстоянии между векторами, равном 8.333, или 0.833/*b*.

Радиальная базисная сеть состоит из двух слоев: скрытого радиального базисного слоя, имеющего S^1 нейронов, и выходного линейного слоя, имеющего S^2 нейронов (рис. 3). Входами блока ||dist|| на этом рисунке являются вектор входа **p** и матрица весов $\mathbf{IW}^{1,1}$, а выходом – вектор, состоящий из S^1 элементов. Выход блока ||dist|| умножается поэлементно на вектор смещения $\mathbf{b}^1$ и формирует вход функции активации.

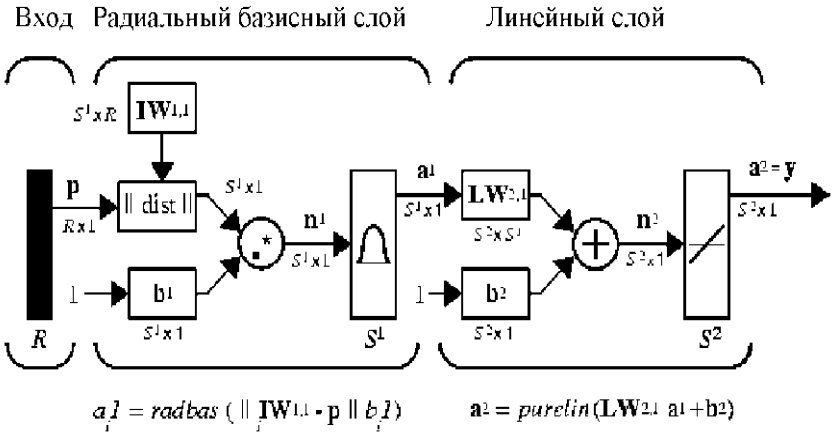


Рис. 3. Радиальная базисная сеть

Все операции, связанные с созданием радиальной базисной сети в системе MATLAB, оформлены в виде специальных функций newrbe и newrb. Первая позволяет построить радиальную базисную сеть с нулевой ошибкой, вторая позволяет управлять количеством нейронов в скрытом слое. Пусть задана функция *f(x)* графически без аналитического описания (рис. 4). Необходимо подобрать такую аналитическую зависимость, которая с заданной точностью описывала бы эту функциональную зависимость. В данном случае используется сумма радиальных базисных функций. Идея аппроксимации может быть представлена графически следующим образом. Рассмотрим взвешенную сумму трех радиальных базисных функций, заданных на интервале
[-3; 3].

```
clear,  p = -3:.1:3;
al = radbas(p);
```

```
a2 = radbas(p - 1.5);
a3 = radbas(p + 2);
a = a1 + a2*1 + a3*0.5;
figure(1),   clf,
plot(p,a1,p,a2,p,a3*0.5,'LineWidth',1.5),hold on,
plot(p,a,'LineWidth',3,'Color',[1/3,2/3,2/3]),grid on,
legend('a1','a2','a3*0.5','a')
```

Как следует из рисунка 4, при задании вектора входа каждый нейрон радиального базисного слоя выдаст значение в соответствии с тем, как близок вектор входа к вектору весов каждого нейрона.

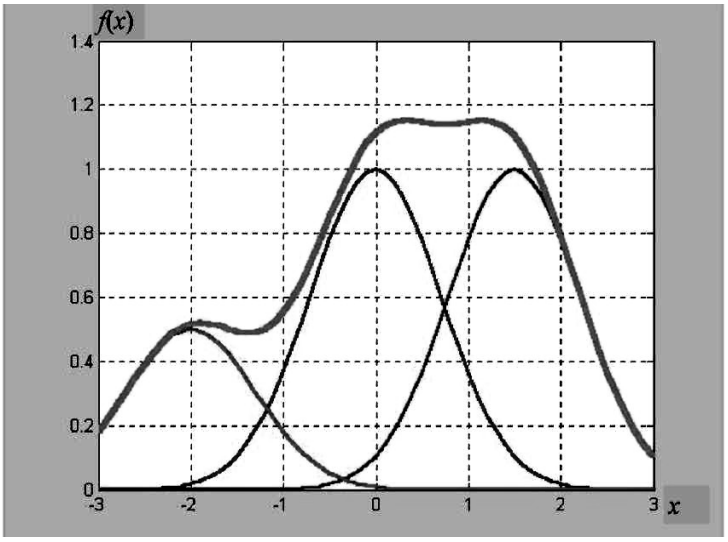


Рис. 4. Функция f(x)

Радиальные базисные нейроны с векторами весов, значительно отличающимися от вектора входа p, будут иметь выходы, близкие к 0, и их влияние на выходы линейных нейронов будет незначительно. Напротив, радиальный базисный нейрон с вектором весов, близким к вектору входа p, выдаст значение, близкое к 1, и это значение будет передано на линейный нейрон с весом, соответствующим выходному слою. Итак, если только один радиальный базисный нейрон имеет выход 1, а все другие имеют выходы, равные или очень близкие к 0, то выход линейного слоя будет равен весу активного выходного нейрона. Однако это исключительный случай. Обычно выход формируют несколько нейронов с разными значениями весов. Выполним ручной расчет для анализа a_1, a_2, a_3.

$$radbas(n)=e^{-n^2}.$$

$$p=2;\ w_1=-2;\ w_2=0;\ w_3=1{,}5;\ b_1=b_2=b_3=1,$$

$$\|p-w_1\|b_1=|2-(-2)|\cdot 1=4,$$

$$\|p-w_2\|b_2=|2-0|\cdot 1=2,$$

$$\|p-w_3\|b_3=|2-1{,}5|\cdot 1=0{,}5,$$

$$radbas(n_1)=e^{-4^2}=e^{-16}¿0,$$

$$radbas(n_2)=e^{-2^2}=e^{-4}¿0{,}000001,$$

$$radbas(n_3)=e^{-0{,}5^2}=e^{-0{,}25}¿0{,}8.$$

В точке p = 2 суммируется практически только выход третьего нейрона. Влияние в этой точке первого и второго нейронов незначительно. Выходы a_1 и a_2 близки к 0. Выход a_3 = 0,8 близок к 1. Анализируя рис. 4, приходим также к выводу, что разложение по радиальным базисным функциям обеспечивает необходимую гладкость. Поэтому их применение для аппроксимации произвольных нелинейных зависимостей вполне оправдано.

Рассмотрим пример формирования радиальной базисной сети в системе MATLAB для решения задачи аппроксимации. Сформируем обучающее множество и зададим допустимое значение функционала ошибки, равное 0.01, параметр влияния определим равным 1 и будем использовать итерационную процедуру формирования радиальной базисной сети:

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405 .6600 .4609 .1336
-.2013 -.4344 -.5000 -.3930 -.1647 .0988 .3072 .3960
     .3449  .1816 -.0312 -.2189 -.3201];
GOAL = 0.01; SPREAD = 1;
net = newrb(P,T,GOAL,SPREAD);% Создание сети
net.layers{1}.size % Число нейронов в скрытом слое
NEWRB, neurons = 0, SSE = 3.69051
ans = 6
```

Для заданных параметров нейронная сеть состоит из 6 нейронов и обеспечивает следующие возможности аппроксимации нелинейных зависимостей после обучения.

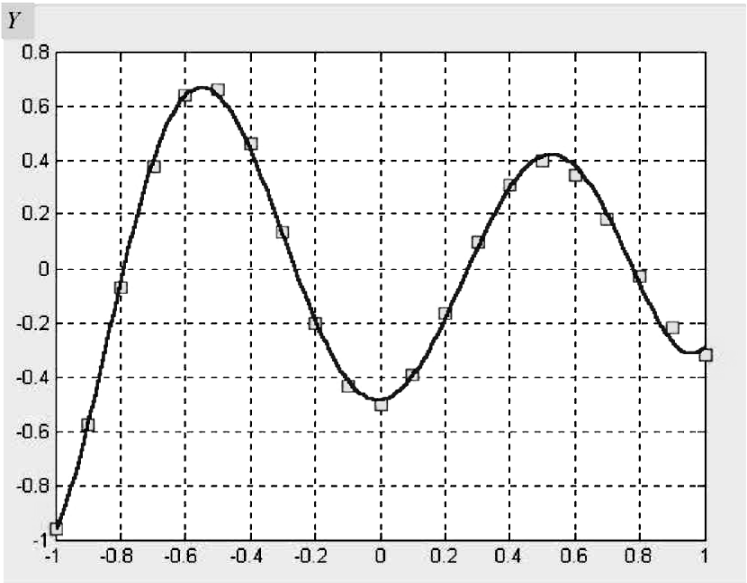


Рис. 5. Аппроксимация нелинейной зависимости

Моделируя сформированную нейронную сеть, построим аппроксимационную кривую на интервале [-1; 1] с шагом 0.01 для нелинейной зависимости.

```
figure(1), clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on;
X = -1:.01:1;
Y = sim(net,X); % Моделирование сети
plot(X,Y,'LineWidth',2), grid on
```

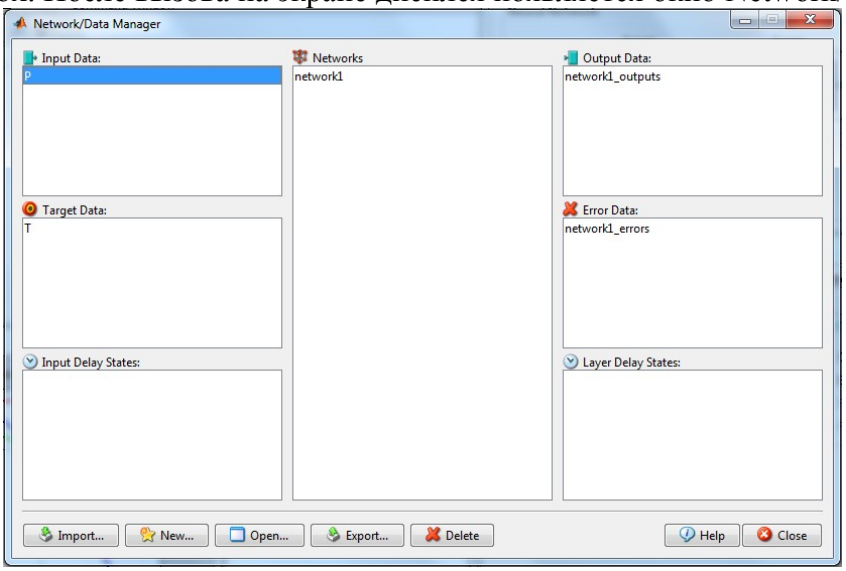
Из анализа рис. 5 следует, что при небольшом количестве нейронов скрытого слоя радиальная базисная сеть достаточно хорошо аппроксимирует нелинейную зависимость, заданную обучающим множеством из 21 точки.

<u>Задание 2.</u> Используя средство NNTool MATLAB, создать и обучить нейронную сеть выполнению операции аппроксимации функции, если заданы последовательности входа и цели.

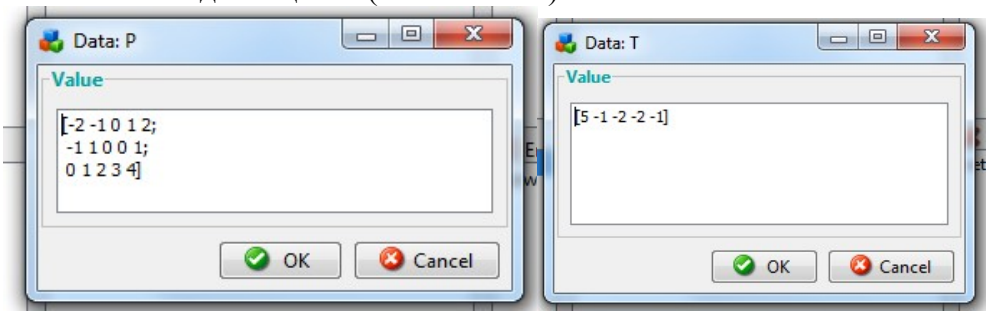
$y=x_1^2-x_2+x_3^2$.

Решение:

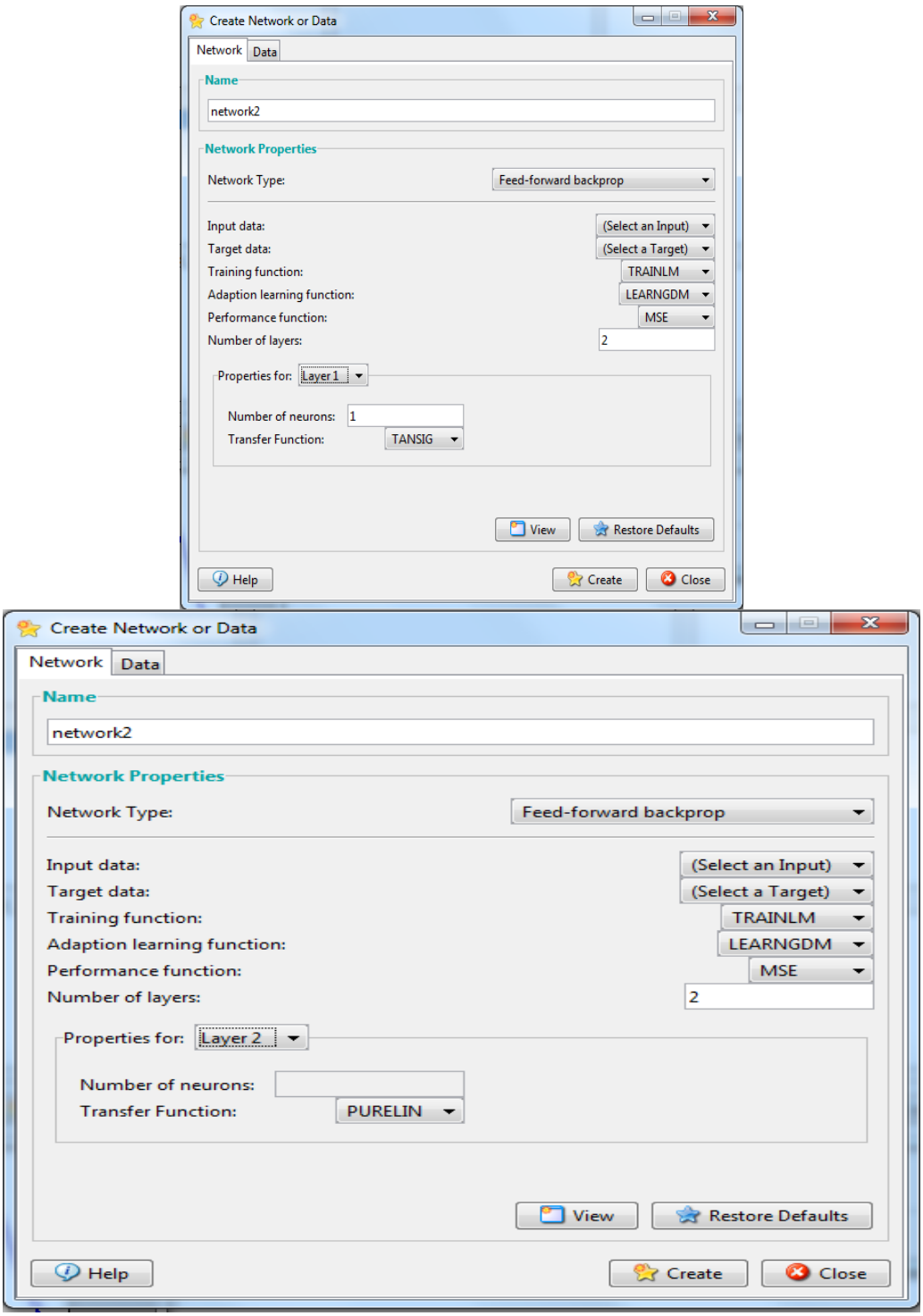
NNTool –инструментальное средство с графическим итерфейсом, позволяющее начинающему пользователю пакета выполнять создание, обучение, моделирование, экспорт и импорт нейронных сетей и данных, не обращаясь к командному окну MATLAB. Вызов NNTool возможен либо с помощью команды nntool из командной строки MATLAB, либо из окна запуска приложений Launch Pad с помощью опции NNTool из раздела Neural Network Toolbox. После вызова на экране дисплея появляется окно Network/Data Manager



Далее нажимаем на кнопку New и появляется окно в котором нам следует сформировать последовательность входов и целей (NEW DATA)

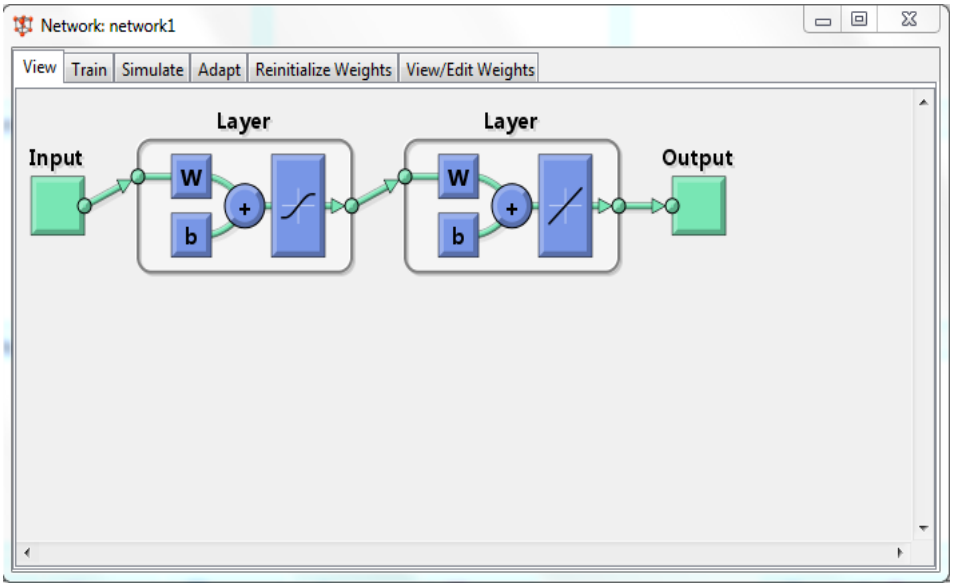


Далее необходимо выбрать тип нейронной сети и задать её параметры

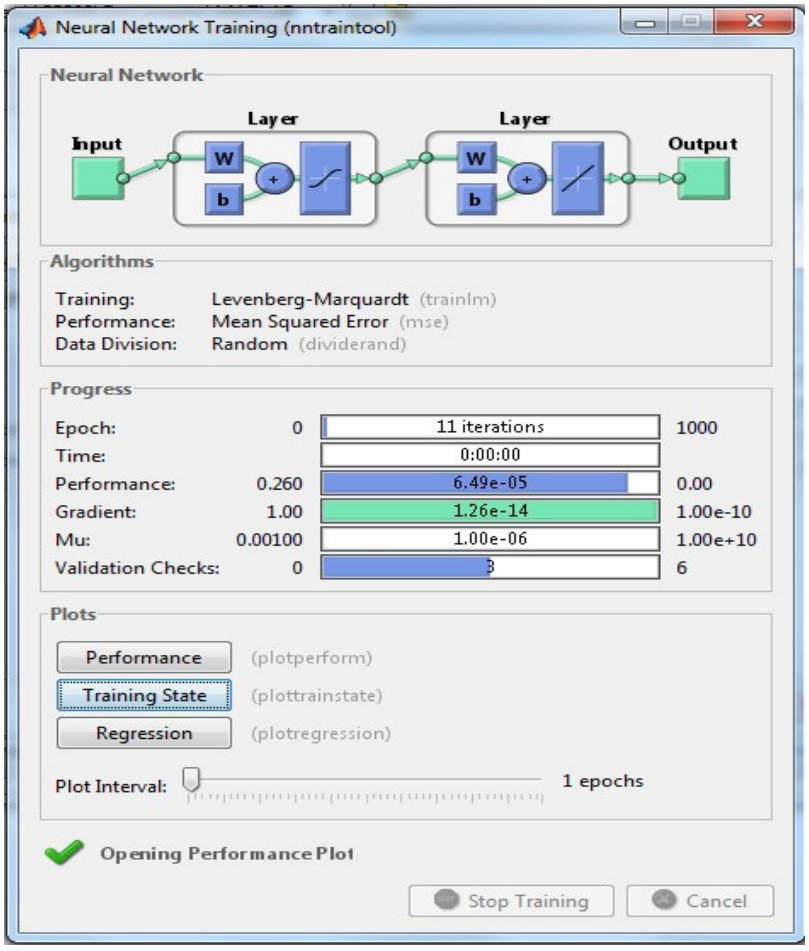


После чего нажать на кнопку CREATE

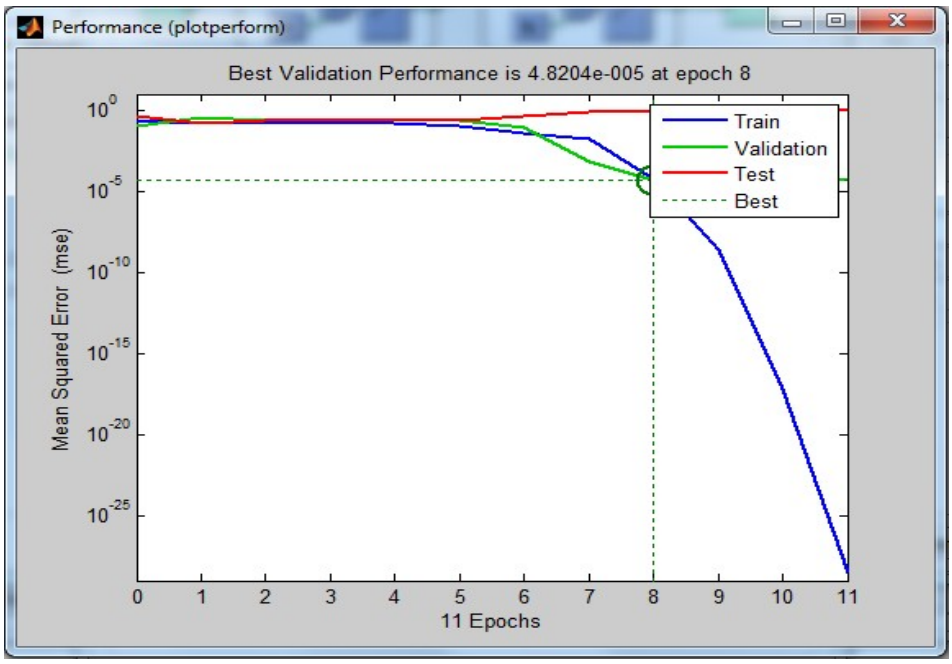
На вкладке View мы можем просмотреть схему нейронной сеть



После чего проводим обучение сети. Приступить к обучению сети можно путем активации кнопки Train Neework.



Далее нажимаем на кнопку Perfomance что бы просмотреть результаты



На графике видно, что после 11 эпохи ошибка стремиться к нулю, следовательно, сеть настроена верно.

<u>Задание 3</u>. Реализовать в MATLAB моделирование персептронной сетью заданной логической функции.

$$\overline{A \to B} \vee \left(\bar{A} \wedge B \right)$$

Для определения функции цели можно воспользоваться таблицей:

A	B	$\bar{A}$	$A \wedge B$	$A \vee B$	$A \to B$	$A \leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Решение:

Персептроны – класс линейных одно- и многослойных нейронных сетей, реализующих прямое распространение сигнала и предназначенных для решения задач классификации линейно отделимых входных векторов. Многослойные персептроны позволяют решать сложные проблемы, связанные с анализом коммутационных соединений, распознаванием образов, и другие задачи классификации с высоким быстродействием и гарантией правильного результата. Нейроны персептронов имеют ступенчатую линию активации hardlim и, следовательно, каждый нейрон возвращает единицу или ноль, что соответствует его возбуждению или торможению.

Определим последовательность двухэлементных векторов входа P для сети с одним двухвходовым нейроном:

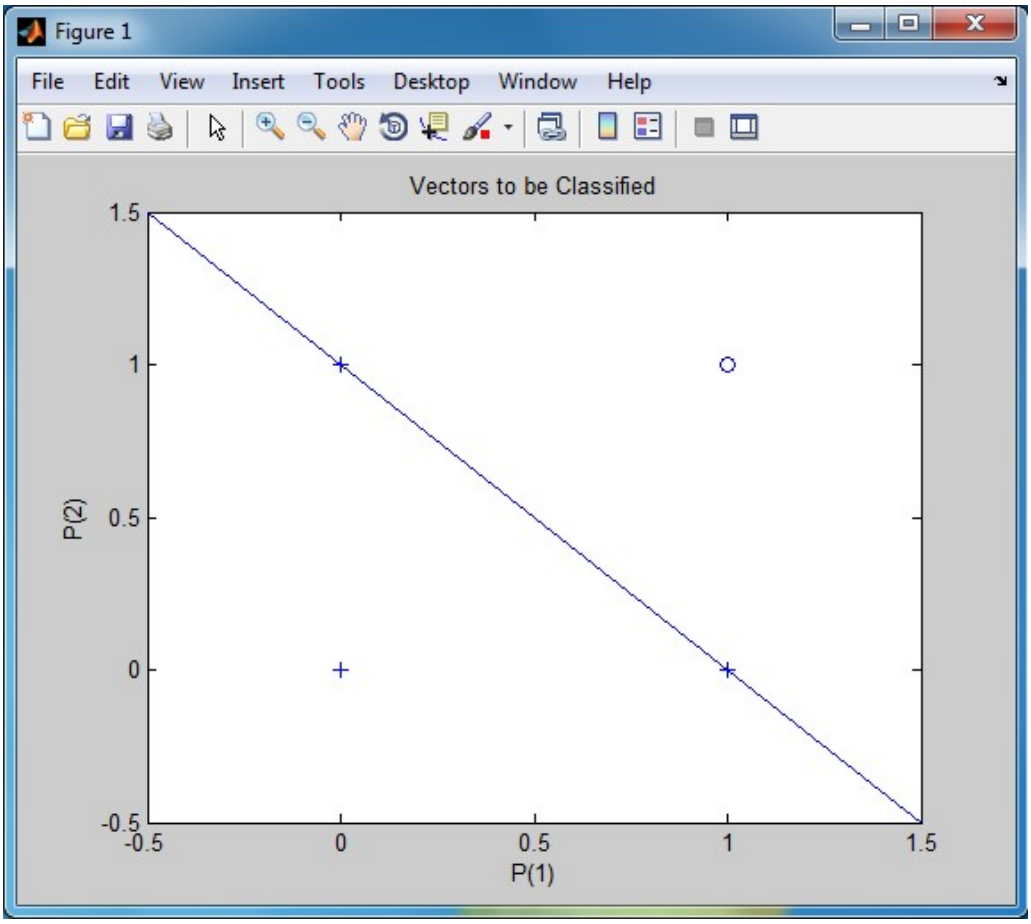
```
>> P=[0 0 1 1;0 1 0 1]
P =
    0    0    1    1
    0    1    0    1
```

Сформируем последовательность целей T:

```
>> T=[1 1 1 0]
T =  1    1    1    0
```

>> net=newp([0 1;0 1],1);
Установим число проходов равным 10 и выполним адаптацию:
>> net.adaptParam.passes=10;
>> net=adapt(net,P,T);

Просмотрим значения весов и смещения, отобразим векторы P и T и построим график разделяющей линии:
>> net.IW{1},net.b{1}
ans = -1 -1
ans = 1
>> plotpv(P,T);
>> linehandle=plotpc(net.IW{1}, net.b{1});
Промоделировать спроектированную нейронную сеть можно с помощью функции sim, подав на вход сети обучающую последовательность, совпадающую с входной последовательностью P:
>> Y=sim(net, P)
Y =1 1 1 0



5. КРИТЕРИИ ОЦЕНИВАНИЯ РАБОТЫ

Оценка «отлично» выставляется студенту, если он продемонстрировал глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы

и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если он продемонстрировал твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если он продемонстрировал твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.

Оценка «неудовлетворительно» выставляется студенту, если он продемонстрировал неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы.

6. ПОРЯДОК ЗАЩИТЫ РАБОТЫ

Защита контрольной работы проводится в виде научного дискоса с демонстрацией выполненных заданий, в соответствии с графиком защиты. После доклада студенту задаются вопросы как преподавателем, так и студентами группы.

В процессе защиты своей работы студент делает доклад продолжительностью 7-10 минут. Доклад должен быть предварительно подготовлен студентом. Студент должен свободно ориентироваться в своей работе.

В выступлении необходимо корректно использовать демонстрационные материалы, которые усиливают доказательность выводов и облегчают восприятие доклада студента.

7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Рекомендуемая литература:

Основная литература:

5. Комарцова Л.Г., Максимов А.В. Нейрокомпьютеры: Учеб. пособие для вузов. – 2-е изд., перераб. и доп. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2013. – 400 с.
6. Медведев В.С., Потемкин В.Г. Нейронные сети. MATLAB 6 / Под общ. ред. В.Г. Потемкина. – М.: ДИАЛОГ-МИФИ, 2012. – 496 с.

11.1.2. Дополнительная литература:

1. Галушкин А.И. Теория нейронных сетей. – М.: ИПРЖР, 2000. – 416 с.
2. Хайкин С. Нейронные сети. Полный курс. – М.: Изд. дом Вильямс, 2006.
3. Чернышев А.Б., Козлов В.А., Жерносек И.А. Нейрокомпьютерные сети: Лабораторный практикум. – Пятигорск: Изд-во ПГТУ, 2011. – 40 с.

11.1.3. Методическая литература:

7. Методические указания по выполнению лабораторных работ по дисциплине «Теория нейронных сетей».
8. Методические указания по выполнению контрольной работы по дисциплине «Теория нейронных сетей».
9. Методические рекомендации для студентов по организации самостоятельной работы по

дисциплине «Теория нейронных сетей».

11.1.4. Интернет-ресурсы:

5. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
6. http://window.edu.ru – образовательные ресурсы ведущих вузов

11.1.5. Программное обеспечение

3. Neural Network Toolbox системы MatLab.