МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____О.Г. Локтионова
«__» _____2017 г.

Моделирование нейросетевых структур в системе MATLAB

Методические указания по лабораторным работам
по дисциплине «Нейросетевые технологии»

Курск 2017

УДК 004.032.26

Составители: Кабус Д.А. Кассим, С.А. Филист

Рецензент
Доктор технических наук, профессор А.Ф. Рыбочкин

Моделирование нейросетевых структур в системе MATLAB: методические указания по лабораторным работам / Юго-Зап. гос. ун-т; сост.: Кабус Д.А. Кассим, С.А. Филист Курск, 2017. 50 с.

Рассмотрены основные конструкции искусственных нейронных сетей и способы их реализации и настройки в системе MATLAB.

Предназначено для студентов направлению подготовки 12.04.04 «Биотехнические системы и технологии».

Текст печатается в авторской редакции

Подписано в печать 5.04.17. Формат 60×84 1/16. Бумага офсетная.
Усл. печ. л. 2,9. Уч.-изд. л. 2,6. Тираж 100 экз. Заказ 539 .
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

Лабораторная работа № 1

Персептроны и однослойные персептронные нейронные сети.

Модель нейрона. Графическая визуализация вычислений

в системе MATLAB

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

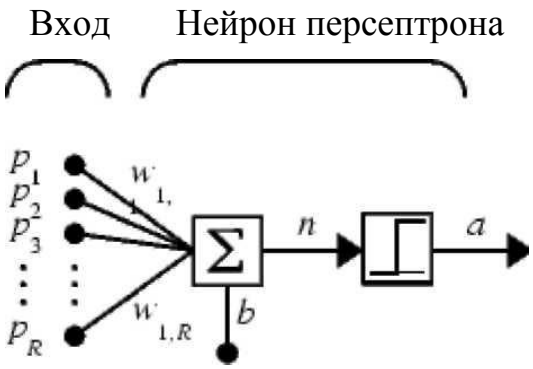
Общие сведения

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

Архитектура персептрона

Нейрон персептрона

Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активации hardlimc жесткими ограничениями (рис. 1).



$$a = \text{hardlim}(\mathbf{Wp} + b)$$

Рис. 1

Каждое значение элемента вектора входа персептрона умножено на соответствующий вес *w*1*j*,и сумма полученных взвешенных элементов является входом функции активации.

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

Функция активации с жесткими ограничениями придает персептрону способность классифицировать векторы входа, разделяя пространство входов на две области, как это показано на рис. 2, для персептрона с двумя входами и смещением.

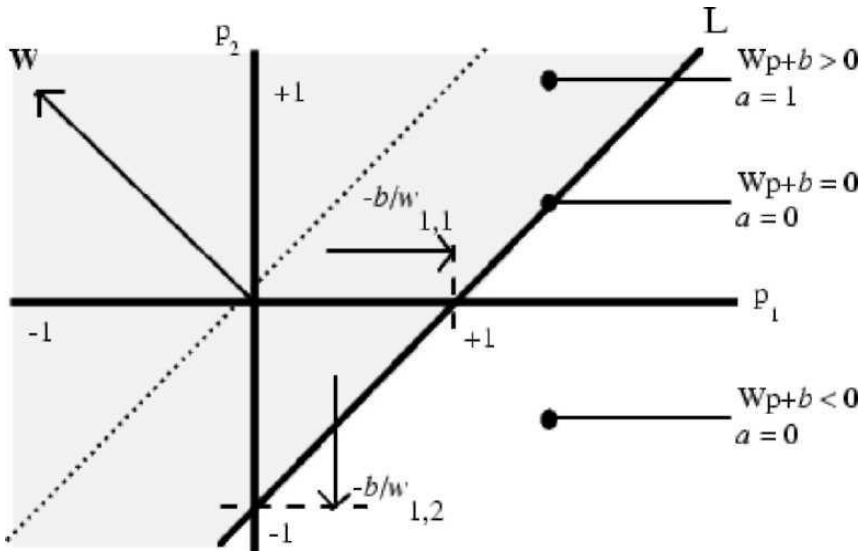


Рис. 2

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

$$\mathbf{W}^{\mathrm{T}}\mathbf{p}+ b= 0. \quad (1)$$

Эта линия перпендикулярна к вектору весов **w**и смещена на величину *b*. Векторы входа выше линии *L*соответствуют положительному потенциалу нейрона, и, следовательно, выход персептрона для этих векторов будет равен 1; векторы входа ниже линии *L*соответствуют выходу персептрона, равному 0.

При изменении значений смещения и весов граница линии *L*изменяет свое положение.

Персептрон без смещения всегда формирует разделяющую линию, проходящую через начало координат; добавление смещения формирует линию, которая не проходит через начало координат, как это показано на рис. 2.

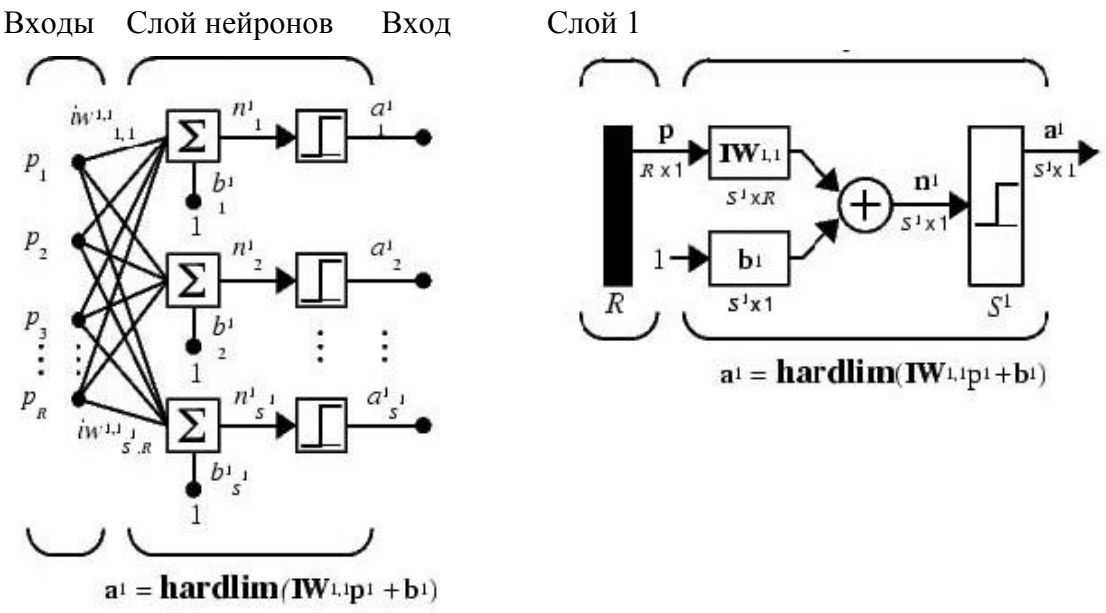
В случае, когда размерность вектора входа превышает 2, т. е. входной вектор **P**имеет более 2 элементов, разделяющей границей будет служить гиперплоскость.

Архитектура сети

Персептрон состоит из единственного слоя, включающего *S*нейронов, как это показано на рис. 3,*а*и *б*в виде развернутой и укрупненной структурных схем соответственно; веса *Wj*- это коэффициенты передачи от j-го входа к i-му нейрону.

Уравнение однослойного персептрона имеет вид

$$\mathbf{a= f\ (Wp+\ b).}$$



а *б*

Рис. 3

Модель персептрона

Для формирования модели однослойного персептрона в системе MATLAB предназначена функция newp

net = newp(**PR**, S)

где **PR**– массив минимальных и максимальных значений для *R*элементов входа размера *R*×2;

S–число нейронов в слое.

Например, функция

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

создает персептрон с одноэлементным входом и одним нейроном; диапазон значений входа – [0 2].

В качестве функции активации персептрона по умолчанию используется функция hardlim.

Моделирование персептрона

Рассмотрим однослойный персептрон с одним двухэлементным вектором входа, значения элементов которого изменяются в диапазоне от –2 до 2 (p_1= [–2 2], p_2= [–2 2], число нейронов в сети S = 1):

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

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

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

Запишем уравнение (1) в развернутом виде для данной сети:

$$\begin{bmatrix} w_{11} \\ w_{12} \end{bmatrix}\begin{bmatrix} p_1 \; p_2 \end{bmatrix} + b_1 = 0$$

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

В этом случае разделяющая линия имеет вид

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

и соответствует линии *L*на рис. 2.

Определим реакцию сети на входные векторы p_1и p_2,расположенные по разные стороны от разделяющей линии:

```
p1 = [1; 1];
a1 = sim(net,p1 % Моделирование сети net с входным
                вектором p1
a1 =
    1
p2 = [1; -1];
a2 = sim(net,p2) % Моделирование сети net с входным
                вектором p2
a2 =
    0
```

Персептрон правильно классифицировал эти два вектора.

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

```
p3 = {[1;  1]  [1; -1]}
a3 = sim(net,p3)  % Моделирование сети netпри
                входномсигнале p3

p3 =
    [2x1double][2x1 double]
a3 =
    [1] [0]
```

Инициализация параметров

Для однослойного персептрона в качестве параметров нейронной сети в общем случае выступают веса входов и смещения. Допустим, что создается персептрон с двухэлементным вектором входа и одним нейроном

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

Запросим характеристики весов входа

```
net.inputweights{1, 1}
ans =
          delays: 0
         initFcn: 'initzero'
           learn: 1
         learnFcn: 'learnp'
       learnParam: []
             size: [1 2]
         userdata: [1x1 struct]
        weightFcn: 'dotprod'
```

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

```
wts = net.IW{1,1},  bias = net.b{1}
wts =
    0    0
bias =
    0
```

Теперь переустановим значения элементов матрицы весов и смещения:

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

```
     5
```

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

```
net = init(net); wts = net.IW{1,1}, bias = net.b{1}
wts =
     0    0
bias =
     0
```

Можно изменить способ, каким инициализируется персептрон с помощью функции init. Для этого достаточно изменить тип функций инициализации, которые применяются для установки первоначальных значений весов входов и смещений. Например, воспользуемся функцией инициализации rands, которая устанавливает случайные значения параметров персептрона:

```
% Задать функции инициализации весов и смещений

net.inputweights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';

% Выполнить инициализацию ранее созданной сети с новыми функциями

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

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

Видно, что веса и смещения выбраны случайным образом.

Порядок выполнения работы

1. Для заданного преподавателем варианта (таблица) разработать структурную схему персептронной нейронной сети.

2. Разработать алгоритм создания и моделирования персептронной нейронной сети.

3. Реализовать разработанный алгоритм в системе MATLAB.

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

5. Построить график, аналогичный представленному на рис. 2, для своих исходных данных.

6. Переустановить значения матриц весов и смещений с помощью рассмотренных функций инициализации.

7. Распечатать текст программы.

8. Составить отчет, который должен содержать :

– цель лабораторной работы;

– структурную схему нейронной сети;

– алгоритм, текст программы и график;

– выводы.

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

Цель работы: изучение структурных схем модели нейрона и средств системы MATLAB, используемых для построения графиков функций активации нейрона.

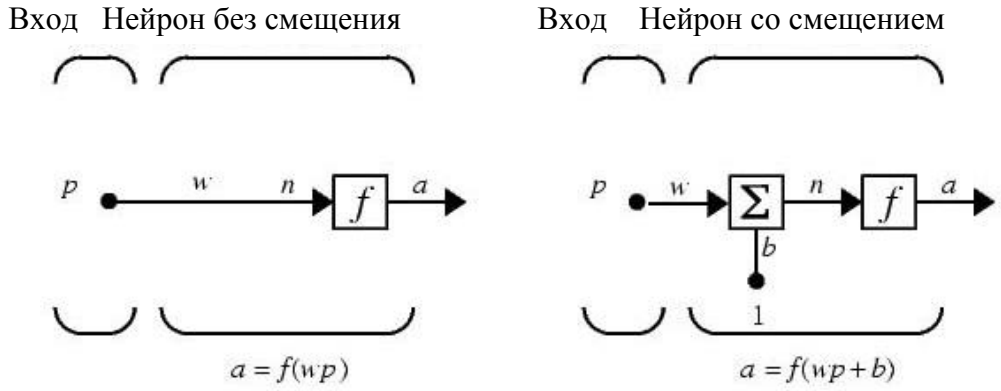
Общие сведения

Простой нейрон

Элементарной ячейкой нейронной сети является *нейрон.* Структура нейрона с единственным скалярным входом показана на рис. 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.

Нейрон, показанный на рис. 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.



аб

Рис. 1

Уравнение нейрона со смещением имеет вид

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

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

$$n = \begin{bmatrix} w & b \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} = w^* p + b^* 1. \quad (2)$$

Нейрон с векторным входом

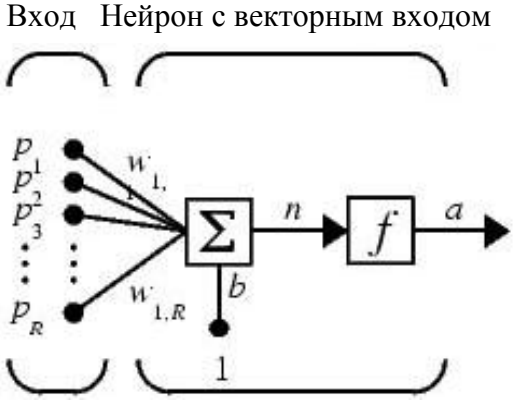
Нейрон с одним вектором входа pс Rэлементами p_1, p_2,..., p_Rпоказан на рис. 2. Здесь каждый элемент входа умножается на веса w_{11}, w_{12},..., w_{1R} соответственно, и взвешенные значения передаются на сумматор. Их сумма равна скалярному произведению вектора-строки **W**на вектор-столбец входа p.

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

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

или

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



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

Рис. 2

и служит аргументом функции активации *f* В нотации языка MATLAB это выражение записывается так:

$$n = \mathbf{W}*\mathbf{p} + b$$

Структура нейрона, показанная выше, является развернутой. При рассмотрении сетей, состоящих из большого числа нейронов, обычно используется укрупненная структурная схема нейрона (рис. 3).

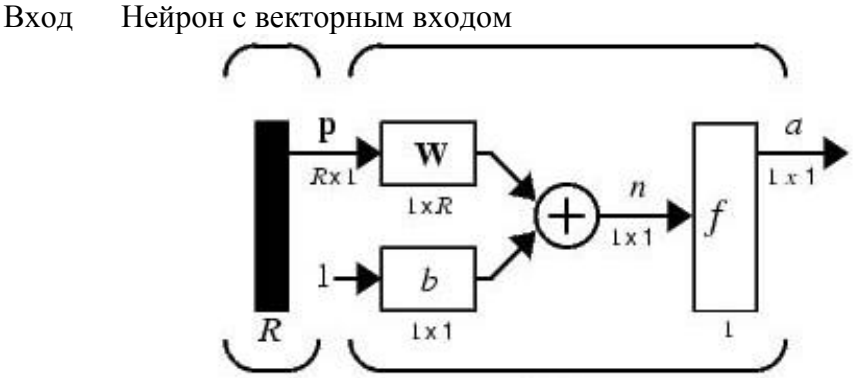


Рис. 2

Вход нейрона изображается в виде темной вертикальной черты, под которой указывается количество элементов входа *R*. Размер вектора входа **p**указывается ниже символа **p**и равен *R*×1.

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 активацииf на выходе которой получаем выход нейрона a, который в данном случае является скалярной величиной.

Структурная схема, приведенная на рис. 3, называется *слоем сети.* Слой характеризуется *матрицей весов* **W,** смещением b, операциями умножения **W*p,** суммирования и функцией активации *f.* Вектор входов **p**обычно не включается в характеристики слоя.

Каждый раз, когда используется сокращенное обозначение сети, размерность матриц указывается под именами векторно-

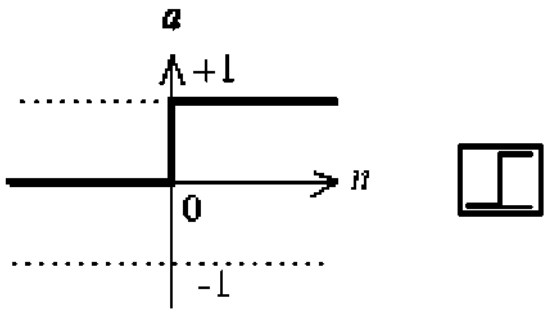
матричных переменных (см. рис. 3). Эта система обозначений поясняет строение сети и связанную с ней матричную математику.

Функции активации

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

Рассмотрим три наиболее распространенные формы функции активации.

Единичная функция активации с жестким ограничением hardlim Эта функция описывается соотношением *a*= hardlim(n) = = 1(n) и показана на рис. 4.



a = hardlim(*n*)

Рис. 4

Она равна 0, если *n*< 0, и равна 1, если *n*≥ 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. 5.

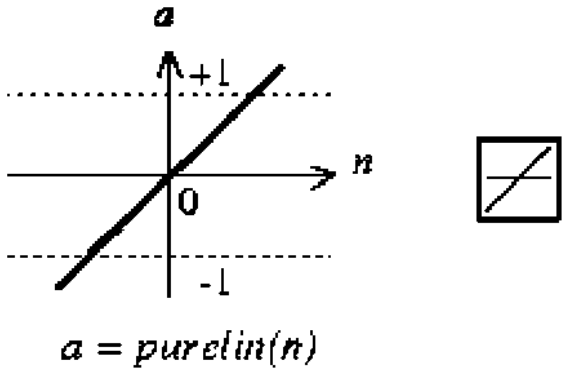


Рис. 5

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

```
16 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.

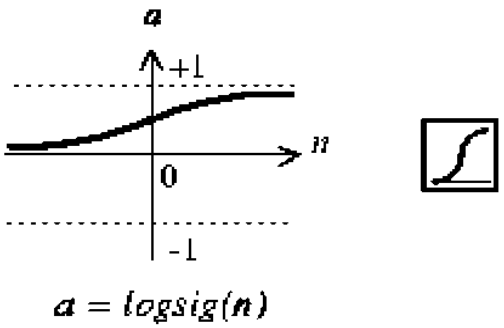


Рис. 2

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+:');
```

На укрупненной структурной схеме для обозначения типа функции активации применяются специальные графические символы; некоторые из них приведены на рис. 7, где *а* – ступенчатая, *б*– линейная, *в* – логистическая.

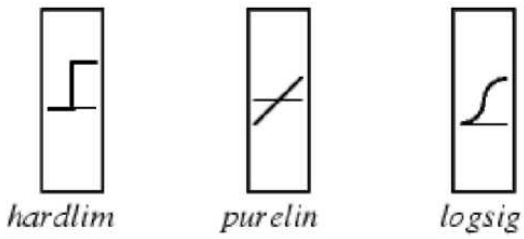


Рис. 2

Построение графиков функций одной переменной в системе MATLAB

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 (рис. 8):

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

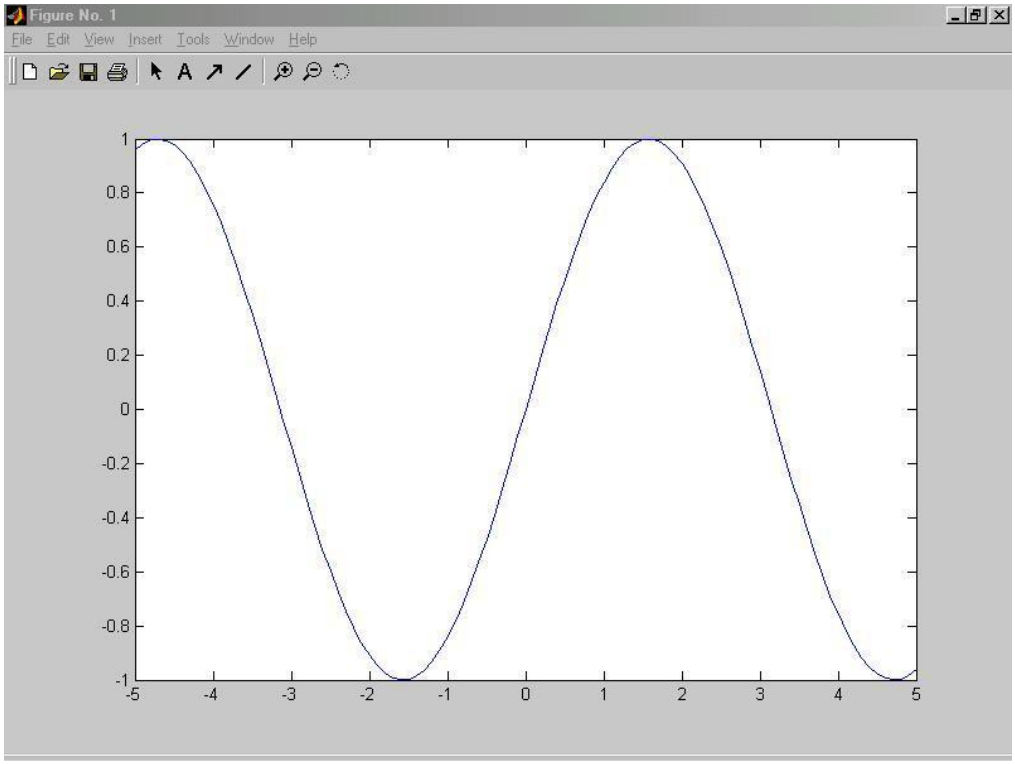


Рис. 8

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 helpplot.

Порядок выполнения работы

1. Построить графики функций активации в заданных диапазонах значений в соответствии с вариантом (таблица), используя функцию plot.

2. Используя функцию plot, построить графики всех заданных функций, согласно варианту, в одном графическом окне.

3. Составить отчет, который должен содержать:

– цель лабораторной работы;

– графики функций;

– выводы.

Номер варианта	Диапазоны значений входа	Имя функции
1	–3...+3	hardlim
2		hardlims
3	–1...+4	purelin
4	–2...+2	poslin
5	–8...+8	satlin
6	–9...+9	satlins
7	–7...+7	radbas
8	–5...+5	tribas
9	–3...+3	logsig
10	–6...+6	tansig

Лабораторная работа № 2

Процедуры настройки параметров персептронных нейронных сетей. Правила настройки. Процедура адаптации

Цель работы: изучение процедуры настройки параметров персептронных нейронных сетей и реализация правил настройки в системе MATLAB.

Общие сведения

Определим процесс обучения персептрона как процедуру настройки весов и смещений с целью уменьшить разность между желаемым (целевым) и истинным сигналами на его выходе, используя некоторое правило настройки (обучения). Процедуры обучения делятся на 2 класса: с учителем и без учителя.

При обучении с учителем задается множество примеров требуемого поведения сети, которое называется обучающим множеством

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\},...,\{\mathbf{p}_Q, \mathbf{t}_Q\}. \quad (1)$$

где $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_Q$–входы персептрона, а $\mathbf{t}_1,\mathbf{t}_2,...,\mathbf{t}_Q$–требуемые (целевые) выходы.

При подаче входных сигналов выходы персептрона сравниваются с целями. Правило обучения используется для настройки весов и смещений персептрона так, чтобы приблизить значения выхода к целевому значению. Алгоритмы, использующие такие правила обучения, называются *алгоритмами обучения с учителем.* Для их успешной реализации необходимы эксперты, которые должны предварительно сформировать обучающие множества. Разработка таких алгоритмов рассматривается как первый шаг в создании систем искусственного интеллекта.

В этой связи ученые не прекращают спора на тему, можно ли считать алгоритмы обучения с учителем естественными и свойственными природе, или они созданы искусственно. Например, обучение человека, на первый взгляд, происходит без учителя: на зри-

тельные, слуховые, тактильные и прочие рецепторы поступает информацияизвне, и внутри мозга происходит некая самоорганизация. Однако нельзя отрицать и того, что в жизни человека немало учителей – и в буквальном, и в переносном смысле, которые координируют реакции на внешние воздействия. Вместе с тем как бы ни развивался спор приверженцев этих двух концепций обучения, представляется, что обе они имеют право на существование. И рассматриваемое нами правило обучения персептрона относится к правилу обучения с учителем.

При обучении без учителя веса и смещения изменяются только в связи с изменениями входов сети. В этом случае целевые выходы в явном виде не задаются. Главная черта, делающая обучение без учителя привлекательным, –это его самоорганизация, обусловленная, как правило, использованием обратных связей. Что касается процесса настройки параметров сети, то он организуется с использованием одних и тех же процедур. Большинство алгоритмов обучения без учителя применяется при решении задач кластеризации данных, когда необходимо разделить входы на конечное число классов.

Правила настройки персептронных нейронных сетей

Настройка параметров (обучение) персептрона осуществляется с использованием обучающего множества. Обозначим через **p**вектор входов персептрона, а через **t**–вектор соответствующих желаемых выходов. Цель обучения –уменьшить погрешность **e = a – t**, которая равна разности между реакцией нейрона **a**и вектором цели **t**.

Правило настройки (обучения) персептрона должно зависеть от величины погрешности **e**.Вектор цели **t**может включать только значения 0 и 1, поскольку персептрон с функцией активации hardlimможет генерировать только такие значения.

При настройке параметров персептрона без смещения и с единственным нейроном возможны только три ситуации:

1. Для данного вектора входа выход персептрона правильный ($a = t$ и $e = t - a = 0$), и тогда вектор весов **w**не претерпевает изменений.

2. Выход персептрона равен 0, а должен быть равен 1 (a= 0, t = 1 и e= t– 0 = 1). В этом случае вход функции активации $\mathbf{w}^{x}\mathbf{p}$отрицательный и его необходимо скорректировать. Добавим к вектору весов $\mathbf{w}$вектор входа $\mathbf{p}$, и тогда произведение $(\mathbf{w}^{T}+ \mathbf{p}^{T})\ \mathbf{p}= \mathbf{w}^{T}\mathbf{p}+ \mathbf{p}^{T}\mathbf{p}$изменится на положительную величину, а после нескольких таких шагов вход функции активации станет положительным и вектор входа будет классифицирован правильно. При этом изменяется настройка весов.

3. Выход персептрона равен 1, а должен быть равен 0 (a = 1, t = 0 и e= t– 0 = –1). В этом случае вход функции активации $\mathbf{w}^{x}\mathbf{p}$положительный и его необходимо скорректировать. Вычтем из вектора весов $\mathbf{w}$вектор входа $\mathbf{p}$, и тогда произведение $(\mathbf{w}^{T}–\mathbf{p}^{T})\ \mathbf{p}= = \mathbf{w}^{T}\mathbf{p}–\mathbf{p}^{T}\mathbf{p}$изменится на отрицательную величину, а после нескольких таких шагов вход функции активации станет отрицательным и вектор входа будет классифицирован правильно. При этом изменяется настройка весов.

Теперь правило настройки (обучения) персептрона можно записать, связав изменение вектора весов $\Delta\mathbf{w}$с погрешностью $e= t– a$:

$$\Delta w = \begin{cases} 0, \text{ если } e = 0; \\ \mathbf{p}^{T}, \text{ если } e = 1; \\ -\mathbf{p}^{T}, \text{ если } e = -1. \end{cases}$$

Все три случая можно описать одним соотношением

$$\Delta\mathbf{w}= (t–a)\mathbf{p}^{T}= e\mathbf{p}^{T}.$$

Можно получить аналогичное выражение для изменения смещения, учитывая, что смещение можно рассматривать как вес для единичного входа:

$$\Delta b= (t–a)\ 1 = e.$$

В случае нескольких нейронов эти соотношения обобщаются следующим образом:

$$\begin{cases} \Delta\mathbf{W}^{T} = (\mathbf{t}-\mathbf{a})\mathbf{p}^{T} \\ \Delta\mathbf{b} = (\mathbf{t}-\mathbf{a}) = \mathbf{e} \end{cases}$$

Тогда правило настройки (обучения) персептрона можно записать в следующей форме:

$$\begin{cases} \mathbf{W}^{\mathrm{T}new} = \mathbf{W}^{\mathrm{T}old} + \mathbf{e}\mathbf{p}^{\mathrm{T}}; \\ \mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}. \end{cases}$$

Описанные соотношения положены в основу алгоритма настройки параметров персептрона, который реализован в ППП NeuralNetworkToolbox в виде функции learnp. Каждый раз при выполнении функции learnp будет происходить перенастройка параметров персептрона, и, если решение существует, то процесс обучения персептрона сходится за конечное число итераций. Если смещение не используется, то функция learnp ищет решение, изменяя только вектор весов **w.** Это приводит к нахождению разделяющей линии, перпендикулярной вектору **w,** которая должным образом разделяет векторы входа.

Рассмотрим простой пример персептрона с единственным нейроном и двухэлементным вектором входа

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

Определим смещение *b*равным 0, а вектор весов **w**равным [1 – 0,8]

```
net.b{1} = 0;
w = [1 -0,8];   net.IW{1,1} = w;
```

Обучающее множество зададим следующим образом:

```
p = [1; 2]; t = [1];
```

Моделируя персептрон, рассчитаем выход и ошибку на первом шаге настройки:

```
a = sim(net,p),   e = t-a
a =
     0
e =
     1
```

Используя функцию настройки параметров learnp, найдем требуемое изменение весов:

```
dw = learnp(w,p,[ ],[ ],[ ],[ ],e,[ ],[ ],[ ])
dw =
     1    2
```

Тогда новый вектор весов примет вид

```
w = w + dw
w =
     2.0000   1.2000
```

Описанные выше правило и алгоритм настройки (обучения) персептрона гарантируют сходимость за конечное число шагов для всех задач, которые могут быть решены с использованием персептрона. Это в первую очередь задачи классификации векторов, которые относятся к классу линейно отделимых, когда все пространство входов можно разделить на две области некоторой прямой линией, в многомерном случае – гиперплоскостью.

Порядок выполнения работы

1. Для заданного преподавателем варианта задания (таблица) выполнить ручной расчет настройки весов и смещений персептронной нейронной сети.

2. Разработать алгоритм создания и моделирования персептрон-ной нейронной сети.

3. Реализовать разработанный алгоритм в системе MATLAB.

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

5. Сравнить результаты ручных расчетов и расчетов, выполненных в системе MATLAB.

6. Распечатать текст программы.

7. Составить отчет, который должен содержать:

– цель лабораторной работы;

– структурную схему нейронной сети;

– алгоритм, текст программы и график;
– ручной расчет параметров нейронной сети;
– выводы.

Номер варианта	Количество входов – 2; количество нейронов – 1		
	Диапазоны значений входов	Входы персептрона	Целевые выходы
1	–1...+4	{[–3; 1] [2; –1] [2; 2] [3; –1]}	{1 1 0 0}
2	–3...+3	{[–2; –1] [1; –1] [0; 1] [2; 0]}	{1 0 1 0}
3	–2...+2	{[0; 0] [1; 1] [–1; 1] [–1; 0]}	{0 0 1 1}
4	–1...+4	{[–2; 2] [1; 2] [0; 0] [3; –2]}	{0 1 0 1}
5	–3...+3	{[–1; 1] [–2; –1] [1; –2] [2; 0]}	{1 0 0 1}
6	–2...+2	{[0; 0] [–1; 1] [–1; 0] [1; 1]}	{0 1 1 0}
7	–1...+4	{[–2; 1] [1; –2] [3; –1] [2; 2]}	{0 0 0 1}
8	–3...+3	{[–2; 1] [0; 1] [2; –1] [–2; –1]}	{0 0 1 0}
9	–2...+2	{[–1; –1] [0; 0] [1; –1] [1; 1]}	{1 1 1 0}
10	–4...+4	{[0; 0] [2; –2] [1; –2] [–2; –1]}	{0 1 0 0}

Цель работы: изучение алгоритма настройки параметров персептронных нейронных сетей с помощью процедуры адаптации в системе MATLAB.

Общие сведения

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

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

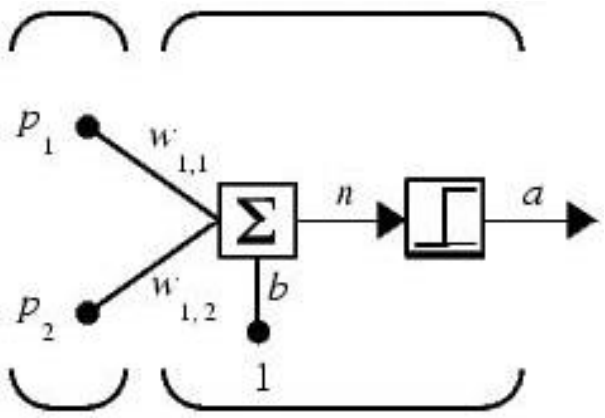
Чтобы пояснить процедуру адаптации, рассмотрим простой пример. Выберем персептрон с одним нейроном и двухэлементным вектором входа (рисунок).

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

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

$$\left\{\mathbf{p}_1=\begin{bmatrix}2\\2\end{bmatrix},t_1=0\right\}\left\{\mathbf{p}_2=\begin{bmatrix}1\\-2\end{bmatrix},t_2=1\right\}\left\{\mathbf{p}_3=\begin{bmatrix}-2\\2\end{bmatrix},t_3=0\right\}\left\{\mathbf{p}_4=\begin{bmatrix}-1\\1\end{bmatrix},t_4=1\right\}. \quad (1)$$

Вход Нейрон персептрона



a = hardlim(**Wp** + *b*)

Рис.

Используем нулевые начальные веса и смещения. Для обозначения переменных каждого шага используем круглые скобки. Таким образом, начальные значения вектора весов $\mathbf{w}^{\mathrm{T}}(0)$ и смещения 6(0) соответственно равны $\mathbf{w}^{\mathrm{T}}(0) = [0\ 0]$ и $b(0) = 0$.

1-й шаг процедуры адаптации

Вычислим выход персептрона для первого вектора входа **p**i, используя начальные веса и смещение:

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

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

$$\begin{cases} e = t_1 - a = 0 - 1 = -1; \\ \Delta\mathbf{w}^{\mathrm{T}} = e\mathbf{p}_1^{\mathrm{T}} = (-1)\left[2\ 2\right] = \left[-2\ -2\right]; \\ \Delta b = e = (-1) = -1 \end{cases} \tag{3}$$

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

$$\begin{cases} \Delta\mathbf{w}^{\mathrm{T}^{new}} = \mathbf{w}^{\mathrm{T}^{old}} + \Delta\mathbf{w}^{\mathrm{T}} = \left[0\ 0\right] + \left[-2\ -2\right] = \left[-2\ -2\right] = \mathbf{w}^{\mathrm{T}}(1); \\ b^{new} = b^{old} + \Delta b = 0 + (-1) = -1 = b(1). \end{cases} \tag{4}$$

*2-й шаг процедуры адаптации*Обратимся к новому вектору входа $\mathbf{p}_2$,тогда

$$a = \text{hard lim}\left(w^m(1)\,p_2 + b(1)\right) = \\ = \text{hard lim}\left(\left[\left[-2\ -2\right]\begin{bmatrix}1\\-2\end{bmatrix}\right] + (-1)\right) = \text{hard lim}(1) = 1 \tag{5}$$

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

$$\begin{cases} \mathbf{w}^{\mathrm{T}}(2) = \mathbf{w}^{\mathrm{T}}(1) = \left[-2\ -2\right]; \\ b(2) = b(1) = -1. \end{cases} \tag{6}$$

3-й шаг процедуры адаптации

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

$$\begin{cases} \mathbf{w}^{\mathrm{T}}(2) = \mathbf{w}^{\mathrm{T}}(2) = [-2 \ \ -2]; \\ b(3) = b(2) = -1. \end{cases} \tag{7}$$

4-й шаг процедуры адаптации

После четвертого примем значение

$$\begin{cases} \mathbf{w}^{\mathrm{T}}(4) = [-3 \ \ -1]; \\ b(4) = 0. \end{cases} \tag{8}$$

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

5-й шаг процедуры адаптации

Вновь используем первый член обучающей последовательности и получаем

$$\begin{cases} \mathbf{w}^{\mathrm{T}}(5) = \mathbf{w}^{\mathrm{T}}(4) = [-3 \ \ -1]; \\ b(5) = b(4) = 0. \end{cases} \tag{9}$$

6-й шаг процедуры адаптации

Переходя ко второму члену, получим следующий результат:

$$\begin{cases} \mathbf{w}^{\mathrm{T}}(6) = [-2 \ \ -3]; \\ b(6) = 1. \end{cases} \tag{10}$$

Этим заканчиваются ручные вычисления.

Расчеты с использованием функции adapt

Вновь сформируем модель персептрона, изображенного на рисунке:

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

Введем первый элемент обучающего множества:

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

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

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

Скорректированные вектор весов и смещение определим следующим образом:

```
twts = net.IW{1,1},  tbiase = net.b{1}
twts =
    -2  -2
tbiase
    -1
```

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

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

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

Теперьобучимсеть.

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

Возвращаются выход и ошибка

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

Скорректированные вектор весов и смещение определяем следующим образом:

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

Моделируя полученную сеть по каждому входу, получим

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

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

Выполним еще один цикл настройки:

```
[net,a,e] = adapt(net,p,t);  a, e
a =
     [0][0][0][1]
e =
     [0]  [1]  [0] [0]
twts = net.IW{1,1},  tbiase = net.b{1} twts =
    -2 -3
tbiase =
     1
a1 = sim(net,p)

a1 =
     [0]  [1]  [0] [1]
```

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

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

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

Нейронные сети на основе персептрона имеют ряд ограничений. Во-первых, выход персептрона может принимать только одно из двух значений (0 или 1); во-вторых, персептроны могут решать задачи классификации только для линейно отделимых наборов векторов. Если векторы входа линейно неотделимы, то процедура адаптации не в состоянии классифицировать все векторы должным образом.

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

Итак, основное назначение персептронов – решать задачи классификации. Они великолепно справляются с задачей классификации линейно отделимых векторов, при этом сходимость гарантируется за конечное число шагов.

Количество циклов обучения зависит от длины отдельных векторов, но и в этом случае решение может быть построено. Демонстрационная программа demop4 поясняет, как влияет выброс длины вектора на продолжительность обучения.

Решение задачи классификации линейно неотделимых векторов возможно либо путем предварительной обработки входных векторов с целью сформировать линейное отделимое множество входных векторов, либо путем использования многослойных персептронов. Возможно также применить другие типы нейронных сетей, например, линейные сети или сети с обратным распростра-

нением, которые могут выполнять классификацию линейно неотделимых векторов входа.

Порядок выполнения работы

1. Для обучающего множества персептронной нейронной сети, разработанной в лабораторной работе № 3, при нулевых начальных значениях весов и смещения выполнить процедуру адаптации ручным расчетом и моделированием с использованием функции adapt системы MATLAB.

2. Определить количество циклов настройки сети. Сравнить результаты расчетов с результатами, полученными в лабораторной работе № 3.

3. Осуществить моделирование настроенной нейронной сети для пяти новых наборов входных векторов и проверить правильность решения задачи классификации сетью.

4. Повторить процедуру настройки персептронной нейронной сети при нулевых начальных значениях весов и смещения с использованием функции adapt системы MATLAB, увеличив длину одного из векторов обучающего множества в 10-30 раз. Сравнить количество циклов обучения с результатами п. 1.

5. Повторить процедуру настройки персептронной нейронной сети при нулевых начальных значениях весов и смещения с использованием функции adapt системы MATLAB, уменьшив длину одного из векторов обучающего множества в 10-15 раз. Сравнить количество циклов обучения с результатами п. 1.

6. Распечатать текст программы.

7. Составить отчет, который должен содержать :

– цель лабораторной работы;

– структурную схему нейронной сети;

– ручной расчет настройки сети:

– текст программы и результаты моделирования;

– выводы.

Лабораторная работа № 3

Обучение линейной сети. Процедура настройки посредством прямого расчета. Применение линейных сетей. Задача классификации векторов

Цель работы: изучение процедуры настройки параметров линейных нейронных сетей посредством прямого расчета в системе MATLAB.

Общие сведения

Линейные сети, как и персептроны, способны решать только линейно отделимые задачи классификации, однако в них используется другое правило обучения, основанное на методе обучения наименьших квадратов, которое является более мощным, чем правило обучения персептрона.

Для заданной линейной сети и соответствующего множества векторов входа и целей можно вычислить вектор выхода сети и сформировать разность между вектором выхода и целевым вектором, которая определит некоторую погрешность.

В процессе обучения сети требуется найти такие значения весов и смещений, чтобы сумма квадратов соответствующих погрешностей была минимальной, поэтому настройка параметров выполняется таким образом, чтобы обеспечить минимум ошибки.

Эта задача разрешима, так как для линейной сети поверхность ошибки как функция входов имеет единственный минимум, и отыскание этого минимума не вызывает трудностей.

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

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\},...,\{\mathbf{p}_Q, \mathbf{t}_Q\}. \quad (1)$$

Требуется минимизировать одну из следующих функций квадратичной ошибки:

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} \left(t(k) - a(k)\right)^2 \qquad (2)$$

или

$$sse = \sum_{k=1}^{Q} e(k)^2 = \sum_{k=1}^{Q} \left(t(k) - a(k)\right)^2, \qquad (3)$$

где *mse* – средняя квадратичная ошибка;

sse – сумма квадратов ошибок.

Процедура настройки

В отличие от многих других сетей настройка линейной сети для заданного обучающего множества может быть выполнена посредством прямого расчета с использованием newlind, т. е. можно построить поверхность ошибки и найти на этой поверхности точку минимума, которая будет соответствовать оптимальным весам и смещениям для данной сети. Проиллюстрируем это на следующем примере.

Предположим, что заданы следующие векторы, принадлежащие обучающему множеству

```
clear,   P = [1 -1.2];   T = [0.5 1];
```

Структурная схема этого линейного нейрона представлена на рис. 1.

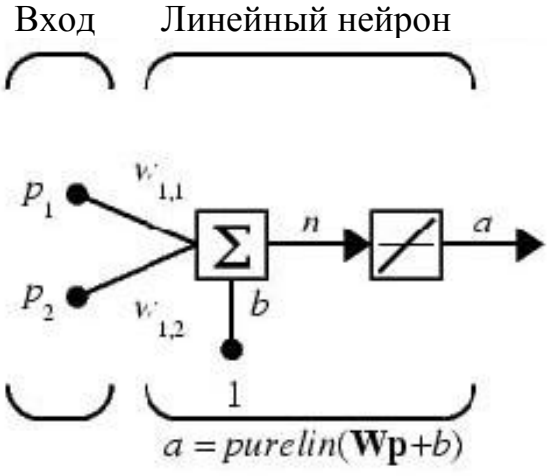


Рис. 1

Запишем уравнение выхода нейрона

$$a = \text{purelin}(n) = \text{purelin}(wp + b) = wp + b. \quad (4)$$

Графическая интерпретация настройки веса и смещения для данного нейрона при двух обучающих множеств сводится к построению прямой, проходящей через две заданные точки и представлена на рис. 2.

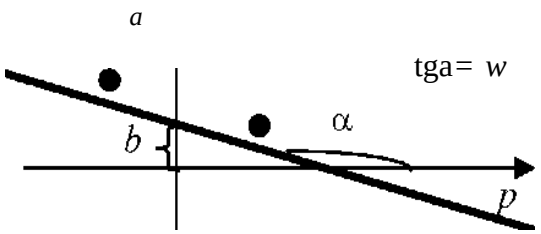


Рис. 2

Построим линейную сеть и промоделируем ее:

```
net = newlind(P,T);
Y = sim(net, P)
Y=
    0.5000 1.0000
net.IW{1,1}
ans =
    -0.2273
net.b
ans =
    [0.7273]
```

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

Зададим следующий диапазон весов и смещений:

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

Рассчитаем критерий качества обучения

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

contour(w range,  b range,ES,20)
hold on
plot(-2.273e-001,7.273e-001, 'x')
holdoff
```

Построим линии уровня поверхности функции критерия качества обучения в пространстве параметров сети (рис. 3):

На графике знаком "x" отмечены оптимальные значения веса и смещения для данной сети.

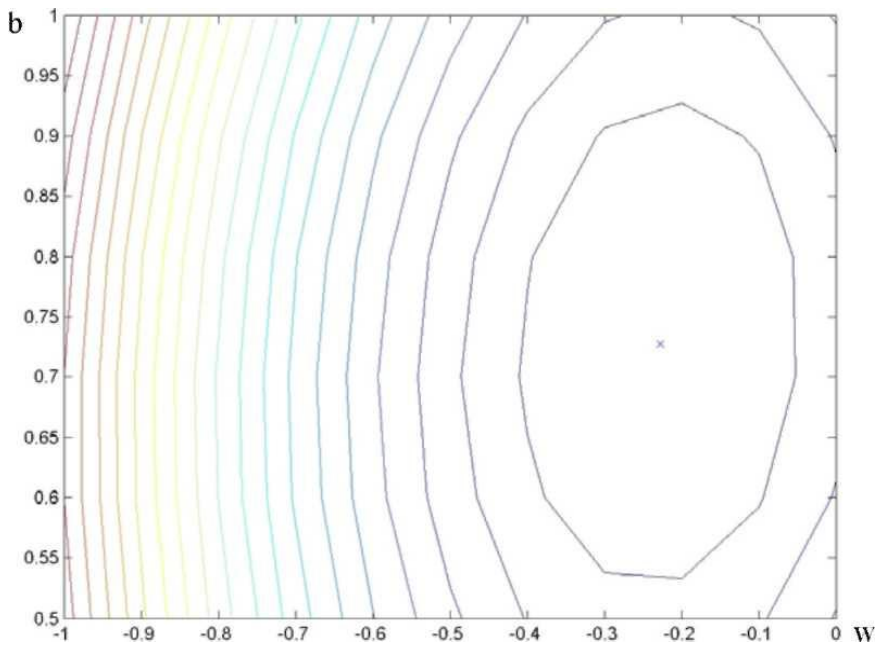


Рис. 3

Порядок выполнения работы

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

2. Построить график для полученных значений веса и смещения, аналогичный рис. 2.

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

4. Сделать ручной расчет значений функции ошибки не менее чем для пяти точек из заданного диапазона.

5. Сравнить результаты ручных расчетов и расчетов, выполненных в системе MATLAB.

6. Распечатать текст программы.

7. Составить отчет, который должен содержать :

– цель лабораторной работы;

– структурную схему нейронной сети;

– алгоритм, текст программы и графики;

– ручной расчет значений функции ошибки и результаты расчета в системе MATLAB;

– выводы.

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

Цель работы: решение задачи классификации с использованием моделей линейной и персептронной нейронных сетей в системе MATLAB.

Общие сведения

Линейные сети могут быть применены для решения задач классификации. Если используется процедура обучения train, то параметры сети настраиваются с учетом суммарного значения функции ошибки. Это отличается от процедуры адаптации adapt, для работы которой характерна настройка параметров с учетом ошибки при представлении каждого вектора входа. Затем обучение применяется к скорректированной сети, вычисляются выходы, сравниваются с соответствующими целями и вновь вычисляется

ошибка обучения.

Если достигнута допустимая погрешность или превышено максимальное число циклов (эпох) обучения, то процедура настройки прекращается.

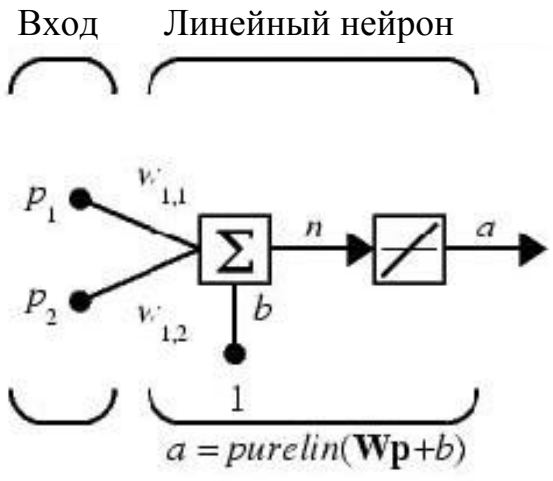
Алгоритм обучения и настройки сходится, если задача классификации разрешима.

Проиллюстрируем решение задачи классификации, ранее решенной с помощью персептрона. Используем для этого простейшую линейную сеть, представленную на рисунке.

$$\left\{\mathbf{p}_1=\begin{bmatrix}2\\2\end{bmatrix},t_1=0\right\}\left\{\mathbf{p}_2=\begin{bmatrix}1\\-2\end{bmatrix},t_2=1\right\}\left\{\mathbf{p}_3=\begin{bmatrix}-2\\2\end{bmatrix},t_3=0\right\}\left\{\mathbf{p}_4=\begin{bmatrix}-1\\1\end{bmatrix},t_4=1\right\}. \quad (1)$$

Обучающее множество представлено следующими четырьмя парами векторов входов и целей:

Определим линейную сеть с начальными значениями веса и смещения, используемыми по умолчанию, т. е. нулевыми; зададим допустимую погрешность обучения, равную 0.1:




```
clear,  p = [2 1 -2 -1;2 -2 2 1]; t = [0 1 0 1];
net = newlin([-2 2; -2 2],1);
```

% Инициализация линейной сети с двумя входами и одним выходом

```
net.trainParam.goal= 0.1;
```

```
[net,   tr]   = train(net,p,t);

TRAINB,  Epoch 0/100,   MSE 0.5/0.1.
TRAINB,  Epoch 25/100,  MSE 0.181122/0.1.
TRAINB,  Epoch 50/100,  MSE 0.111233/0.1.
TRAINB,  Epoch 64/100,  MSE 0.0999066/0.1
TRAINB,  Performancegoalmet.
```

Пороговое значение функции качества достигается за 64 цикла обучения, а соответствующие параметры сети принимают значения:

```
weights = net.iw{1,1},  bias = net.b(1)

weights =
    -0.0615   -0.2194
bias =
    [0.5899]
```

Выполним моделирование созданной сети с векторами входа из обучающего множества и вычислим ошибки сети:

```
A = sim(net, p)

err = t - sim(net,p)

A =
    0.0282   0.9672   0.2741   0.4320
err =
    -0.0282  0.0328   -0.2741  0.56800
```

Заметим, что погрешности сети весьма значительны. Попытка задать большую точность в данном случае не приводит к цели, поскольку возможности линейной сети ограничены. Демонстрационный пример demolin4 иллюстрирует проблему линейной зависимости векторов, которая свойственна и этому случаю.

Порядок выполнения работы

1. Для заданного преподавателем варианта задания (таблица) построить линейную нейронную сеть в системе MATLAB и с ее помощью решить задачу классификации линейно разделимых векторов с точностью 0,01 и максимальным числом эпох 200.

2. Выполнить моделирование созданной линейной сети с векторами входа из обучающего множества и вычислить ошибки сети.

3. Построить персептронную нейронную сеть в системе MATLAB для того же обучающего множества и с ее помощью решить задачу классификации линейно разделимых векторов.

4. Выполнить моделирование созданной персептронной сети с векторами входа из обучающего множества и вычислить ошибки сети.

5. Сравнить результаты моделирования линейной и персептрон-ной линейными сетями.

6. Добавить в обучающее множество такой вектор, чтобы образовались линейно неразделимые векторы. Построить линейную и пер-септронную нейронные сети для решения задачи классификации нового обучающего множества.

7. Выполнить моделирование созданных сетей с векторами входа из обучающего множества и проверить правильность работы сетей.

8. Распечатать текст программы.

9. Составить отчет, который должен содержать :

– цель лабораторной работы;

– структурную схему нейронной сети;

– ручной расчет настройки сети;

– текст программы и результаты моделирования;

– выводы.

Номер варианта	Количество входов – 2; количество нейронов – 1		
	Диапазоны значений входов	Значения входов	Целевые выходы
1	–2...+2	{[–1; –1] [0; 0] [1; –1] [1; 1]}	{1 1 1 0}
2	–1...+4	{[0; 0] [2; –2] [1; –2] [–2; –1]}	{0 1 0 0}
3	–2...+2	{[0; 0] [–1; 1] [–1; 0] [1; 1]}	{0 1 1 0}
4	–1...+4	{[–2; 1] [1; –2] [3; –1] [2; 2]}	{0 0 0 1}
5	–3...+3	{[–2; 1] [0; 1] [2; –1] [–2; –1]}	{0 0 1 0}
6	–2...+2	{[0; 0] [1; 1] [–1; 1] [–1; 0]}	{0 0 1 1}
7	–1...+4	{[–2; 2] [1; 2] [0; 0] [3; –2]}	{0 1 0 1}
8	–3...+3	{[–1; 1] [–2; –1] [1; –2] [2; 0]}	{1 0 0 1}
9	–1...+4	{[–3; 1] [2; –1] [2; 2] [3; –1]}	{1 1 0 0}
10	–3...+3	{[–2; –1] [1; –1] [0; 1] [2; 0]}	{1 0 1 0}

Лабораторная работа № 4

Радиальные базисные сети и их архитектура

Цель работы: изучение модели нейрона и архитектуры радиальной базисной сети; создание и исследование моделей радиальных базисных сетей с нулевой ошибкой в системе MATLAB.

Общие сведения

Создание радиальной базисной сети с нулевой ошибкой

Для построения радиальных базисных сетей с нулевой ошибкой предназначена функция newrbe, которая вызывается следующим образом:

net = newrbe(P, T, SPREAD).

Входными аргументами функции newrbe являются массивы входных векторов **P** и целей **T**, а также параметр влияния SPREAD. Данная функция возвращает радиальную базисную сеть с такими весами и смещениями, что ее выходы точно равны целям **T**, и создает столько нейронов радиального базисного слоя, сколько имеется входных векторов в массиве **P**, и устанавливает веса первого слоя равными $\mathbf{P}^{\mathrm{T}}$ При этом смещения устанавливаются равными 0.8326/ SPREAD. Это означает, что все входы в диапазоне ± SPREAD считаются значимыми, т. е. чем больший диапазон входных значений должен быть принят во внимание, тем большее значение параметра влияния SPREAD должно быть установлено. Это наиболее наглядно проявляется при решении задач аппроксимации функций.

Веса второго слоя $\mathbf{LW}^{2,1}$ и смещений $\mathbf{b}^2$ определяются моделированием выходов первого слоя a1 и решением системы линейных алгебраических уравнений (СЛАУ). Данная СЛАУ в общем виде для Q пар вход (**P**) / цель (**T**) состоит из Q уравнений с $Q + 1$ неизвестными.

Следовательно, имеет одну свободную переменную и характеризуется бесконечным числом решений с нулевой погрешностью.

Рассмотрим пример создания и моделирования следующей радиальной базисной сети.

Пусть дано **p** = [–1 0 1]; **t** = [–0,96 –0,5 –0,32]; **w** = [–1 0 1]

$b_1^1 = b_2^1 = b_3^1 = 0{,}8326$; SPREAD = 1.

Составим СЛАУ для этих данных:

$$a_1^1(-1)w_{21} + a_2^1(-1)w_{22} + a_3^1(-1)w_{23} + b^2 = -0{,}96;$$

$$a_1^1(0)w_{21} + a_2^1(0)w_{22} + a_3^1(0)w_{23} + b^2 = -0{,}5;$$

$$a_1^1(1)w_{21} + a_2^1(1)w_{22} + a_3^1(1)w_{23} + b^2 = -0{,}32;$$

Рассчитаем значения a2; a3 и подставим их в данную систему. Получим

$$1w_{21} + 0{,}5w_{22} + 0{,}0621w_{23} + b^2 = -0{,}96;$$

$$0{,}5w_{21} + 1w_{22} + 0{,}5w_{23} + b^2 = -0{,}5;$$

$$0{,}0621w_{21} + 0{,}5w_{22} + 1w_{23} + b^2 = -0{,}32;$$

т. е. получили три уравнения с четырьмя неизвестными. Принимаем $w_{21} = 0$ и решаем СЛАУ.

В результате находим

$$b_2 = -1{,}163; \quad w_{21} = 0; \quad w_{22} = 0{,}3229; \quad w_{23} = -0{,}6829.$$

Промоделируем пример в системе MATLAB:

```
elear, P = -l:l:l;

T = [-.96 -.50 -.32]; % Создание сети

net = newrbe(P,T);  % Создание радиальной базисной
сети

net.layers{1}.size % Число нейронов в скрытом слое

ans =
     3
net.LW{2,1}
ans =
     0
     0.3229
     -0.6829
net.b{2}
```

```
ans =
    -1.163
```

Порядок выполнения работы

1. Для заданных преподавателем параметров радиальной базисной нейронной сети (таблица) подготовить массивы входных векторов **P** и целей **T** для радиальной базисной нейронной сети с нулевой ошибкой.

2. Разработать структурную схему радиальной базисной нейронной сети с нулевой ошибкой.

3. Составить и решить СЛАУ для разрабатываемой нейронной сети.

4. Создать полученную радиальную базисную сеть в системе MATLAB.

5. Определить параметры созданной нейронной сети (количество нейронов в каждом слое, веса и смещения нейронов).

6. Составить отчет, который должен содержать:

– цель лабораторной работы;

– массивы входных векторов **P** и целей **T**;

– структурную схему нейронной сети;

– СЛАУ для определения параметров выходного слоя;

– текст программы;

– выводы.

Номер варианта	Значения вектора входа	Значения целевого вектора
1	{–2 –1 0 1 2}	{3 1 2 0 –1}
2	{–2 –1 0 1 2}	{–3 –1 2 1 2}
3	{–2 –1 0 1 2}	{–5 1 0 1 –2}
4	{–2 –1 0 1 2}	{–2 1 0 1 –3}
5	{–2 –1 0 1 2}	{3 –1 0 3 2}
6	{–2 –1 0 1 2}	{2 –1 1 –1 2}
7	{–2 –1 0 1 2}	{–2 1 –1 2 1}

8	{–2 –1 0 1 2}	{–3 –1 2 0 3}
9	{–2 –1 0 1 2}	{3 –3 2 2 1}
10	{–2 –1 0 1 2}	{–3 –2 3 –2 2}

Лабораторная работа № 5

Сети PNN

Цель работы: создание и исследование моделей сети PNN в системе MATLAB.

Общие сведения

Нейронные сети PNN (ProbabilisticNeuralNetwork) предназначены для решения вероятностных задач и, в частности, задач классификации.

Архитектура сети

Архитектура сети PNN базируется на архитектуре радиальной базисной сети, но в качестве второго слоя использует так называемый конкурирующий слой, который подсчитывает вероятность принадлежности входного вектора к тому или иному классу и в конечном счете сопоставляет вектор с тем классом, вероятность принадлежности к которому выше. Структура сети PNN представлена на рис. 1.

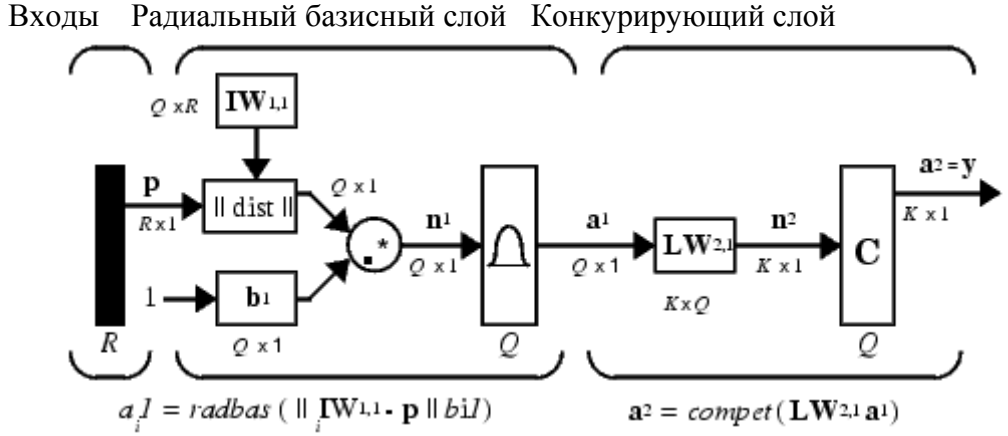


Рис. 1

Предполагается, что задано обучающее множество, состоящее из Q пар векторов вход/цель. Каждый вектор цели имеет K элементов, указывающих класс принадлежности, и, таким образом, каждый вектор входа ставится в соответствие одному из K классов. В результате может быть образована матрица связности **T** размера

K×*Q*, состоящая из нулей и единиц, строки которой соответствуют классам принадлежности, а столбцы – векторам входа. Таким образом, если элемент **T**(*i*,*j*) матрицы связности равен 1, то это означает, что *j*-й входной вектор принадлежит к классу /.

Весовая матрица первого слоя $\mathbf{IW}^{11}$ (net.IW{1,1}) формируется с использованием векторов входа из обучающего множества в виде матрицы $\mathbf{P}^{T}$. Когда подается новый вход, блок ||dist|| вычисляет близость нового вектора к векторам обучающего множества; затем вычисленные расстояния умножаются на смещения и подаются на вход функции активации radbas. Вектор обучающего множества, наиболее близкий к вектору входа, будет представлен в векторе выхода $\mathbf{a}^{1}$ числом близким к 1.

Весовая матрица второго слоя $\mathbf{LW}^{21}$ (net.LW{2,1}) соответствует матрице связности T, построенной для данной обучающей последовательности. Эта операция может быть выполнена с помощью функции ind2vec, которая преобразует вектор целей в матрицу связности **T**. Произведение $\mathbf{T}^{*}\mathbf{a}^{1}$ определяет элементы вектора $\mathbf{a}^{1}$, соответствующие каждому из К классов. В результате конкурирующая функция активации второго слоя compet формирует на выходе значение, равное 1 для самого большего по величине элемента вектора $\mathbf{n}^{2}$ и 0 в остальных случаях. Таким образом, сеть PNN выполняет классификацию векторов входа по *K* классам.

Синтез сети

Для создания нейронной сети PNN предназначена функция newpnn. Определим 7 следующих векторов входа и соотнесем каждый из них к одному из 3 классов:

```
P = [0 0;1 1;0 3;1 4;3 1;4 1;4 3];
Tc = [1 1 2 2 3 3 3].
```

Ставится задача определения: к какому классу принадлежит данный вектор.

Запишем матрицу связности:

```
1 1 0 0 0 0 0
T = 0 0 1 1 0 0 0
0 0 0 0 1 1 1
```

Первая строка – 1 класс; вторая – 2 класс; третья – 3 класс.

В данном случае имеем двумерный вектор (функция от двух переменных).

Графически эта функция представлена на рис. 2.

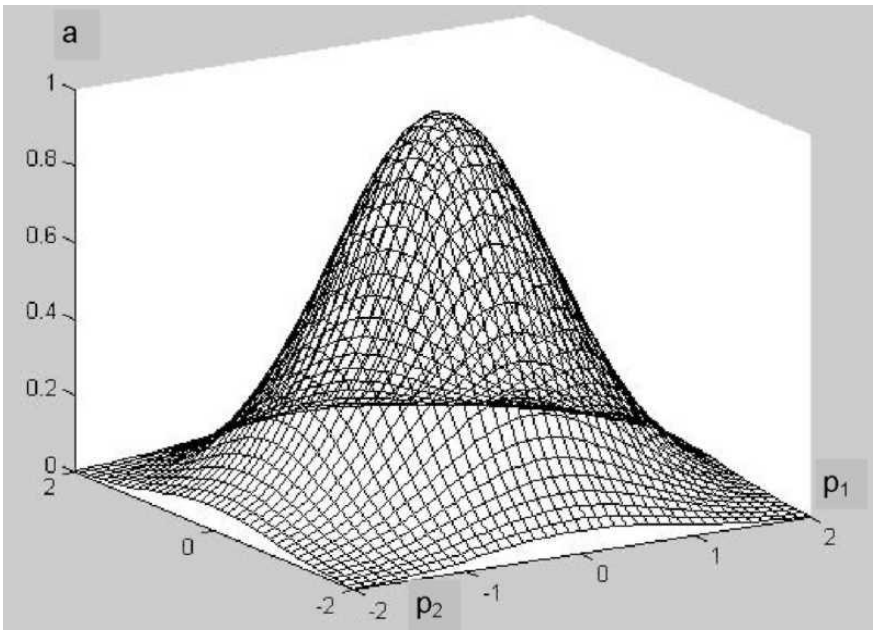


Рис. 2

Найдем выходы первого слоя по формуле

$$a = e^{-\left(\sqrt{(p_1-w_1)^2+(p_2-w_2)^2 b}\right)^2}.$$

Подадим на все нейроны первый обучающий вектор $p_1 = 0$; $p_2 = 0$.

Тогда на выходах имеем:

$$a_1^1 = 1;\ \ a_2^1 = 0,25;\ \ a_3^1 = 0,02;\ \ a_4^1 = 0;\ \ a_5^1 = 0,001;\ \ a_6^1 = 0;\ \ a_7^1 = 0.$$

Веса первого слоя:

$$w_{11}^1 = 0;\ \ w_{12}^1 = 1;\ \ w_{13}^1 = 0;\ \ w_{14}^1 = 1;\ \ w_{15}^1 = 3;\ \ w_{16}^1 = 4;\ \ w_{17}^1 = 4;$$

$$w_{21}^1 = 0;\ \ w_{22}^1 = 1;\ \ w_{23}^1 = 3;\ \ w_{24}^1 = 4;\ \ w_{25}^1 = 1;\ \ w_{26}^1 = 1;\ \ w_{27}^1 = 3.$$

Подадим на все нейронывторой обучающий вектор $p_1 = 0$; $p_2 = 0$.

Тогда на выходах имеем:

$$a_1^1 = e^{-\left(\sqrt{(1-0)^2+(1-0)^2 b}\right)^2};$$

$$a_2^1 = e^{-\left(\sqrt{(1-1)^2+(1-1)^2 b}\right)^2};$$

$$a_3^1 = e^{-\left(\sqrt{(1-0)^2+(1-3)^2 b}\right)^2};$$

$$a_4^1 = e^{-\left(\sqrt{(1-1)^2+(1-4)^2 b}\right)^2};$$

$$a_5^1 = e^{-\left(\sqrt{(1-3)^2+(1-1)^2 b}\right)^2};$$

$$a_6^1 = e^{-\left(\sqrt{(1-4)^2+(1-1)^2 b}\right)^2};$$

$$a_7^1 = e^{-\left(\sqrt{(1-4)^2+(1-3)^2 b}\right)^2};$$

Весам второго слоя присваивают значения разреженной матрицы:

$$w_{11}^2 = 1;\ \ w_{12}^2 = 1;\ \ w_{13}^2 = 0;\ \ w_{14}^2 = 0;\ \ w_{15}^2 = 0;\ \ w_{16}^2 = 0;\ \ w_{17}^2 = 0;$$

$$w_{21}^2 = 0;\ \ w_{22}^2 = 0;\ \ w_{23}^2 = 1;\ \ w_{24}^2 = 1;\ \ w_{25}^2 = 0;\ \ w_{26}^2 = 0;\ \ w_{27}^2 = 0;$$

$$w_{31}^2 = 0;\ \ w_{32}^2 = 0;\ \ w_{33}^2 = 0;\ \ w_{34}^2 = 0;\ \ w_{35}^2 = 1;\ \ w_{36}^2 = 1;\ \ w_{37}^2 = 1.$$

Тогда на выходе 2-го слоя имеем:

$a^2 = \text{compet}(\text{LW}^{2,1} a^1)$;

$$\begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} & W_{15} & W_{16} & W_{17} \\ W_{21} & W_{22} & W_{23} & W_{24} & W_{25} & W_{26} & W_{27} \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} & W_{36} & W_{37} \end{bmatrix} \begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \\ a_5^1 \\ a_6^1 \\ a_7^1 \end{bmatrix}$$

Подставляем значения

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0,25 \\ 0,02 \\ 0 \\ 0,001 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1,25 \\ 0,02 \\ 0,001 \end{bmatrix}$$

получаем:

$$a^2 = \text{compet} \begin{bmatrix} 1,25 \\ 0,02 \\ 0,001 \end{bmatrix}$$

Итак, данный вектор принадлежит к 1-му классу.

На этом ручной расчет закончен.

Перейдем к созданию и исследованию модели в системе MATLAB:

```
clear, P = [0 0;1 1;0 3;1 4;3 1;4 1;4 3]';
Tc = [1 1 2 2 3 3 3];
```

Вектору **Tc** поставим в соответствие матрицу связности **T** в виде разреженной матрицы вида

```
T = ind2vec(Tc)
T =
     (1,1)         1
     (1,2)         1
     (2,3)         1
     (2,4)         1
     (3,5)         1
     (3,6)         1
     (3,7)         1
```

которая определяет принадлежность первых двух векторов классу 1, двух последующих – классу 2 и трех последних – классу 3. Полная матрица**T** имеет вид

```
T = full(T)
T =
     1    1    0    0    0    0    0
     0    0    1    1    0    0    0
     0    0    0    0    1    1    1
```

Массивы P и T задают обучающее множество, что позволяет выполнить формирование сети, промоделировать ее, используя массив входов P, и удостовериться, что сеть правильно решает задачу классификации на элементах обучающего множества. В результате моделирования сети формируется матрица связности, соответствующая массиву векторов входа. Функция vec2ind предназначена для преобразования матрицы связности в индексный вектор:

```
net = newpnn(P,T);
net.layers{1}.size % Число нейронов в сети PNN
ans =
     7
Y = sim(net,P); Yc = vec2ind(Y)
Yc =
     1    1    2    2    3    3    3
```

Результат подтверждает правильность решения задачи классификации.

Теперь выполним классификацию некоторого набора произвольных векторов **p**, не принадлежащих обучающему множеству, используя ранее созданную сеть PNN:

```
p = [1 3; 0 1; 5 2]';
```

Осуществляя моделирование сети для этого набора векторов, получаем

```
a = sim(net,p);  ac = vec2ind(a)
ac =
     2    1    3
```

Порядок выполнения работы

1. Для заданных преподавателем параметров радиальной базисной нейронной сети (таблица) подготовить массивы входных векторов **P** и целей**T** для нейронной сети PNN.
2. Разработать структурную схему нейронной сети PNN.
3. Выполнить ручной расчет определения принадлежности к классу всех векторов из обучающего множества.
4. Создать полученную нейронную сеть в системе MATLAB и сравнить полученные результаты с ручным расчетом.
5. Определить параметры созданной нейронной сети (количество нейронов в каждом слое, веса и смещения нейронов).
6. Выполнить классификацию набора из трех произвольных входных векторов, не принадлежащих обучающему множеству, используя созданную сеть.
7. Составить отчет, который должен содержать:

– цель лабораторной работы;

– массивы входных векторов P и целей T;

– структурную схему нейронной сети;

– ручной расчет;

– текст и результаты работы программы;

– выводы.

Номер варианта	Значения векторов входа	Значения вектора индексов классов
1	[1 3; 2 4; 1 1; 0 0; 4 2; 3 1; 5 3]	[1 1 2 2 3 3 3]
2	[3 2; 4 1; 1 5; 2 4; 0 3; 1 1; 2 2]	[3 3 2 2 2 1 1]
3	[2 5; 1 3; 1 2; 0 1; 1 0; 3 2; 4 1]	[2 2 1 1 1 3 3]
4	[0 1; 1 0; 2 2; 4 2; 5 3; 2 4; 1 5]	[1 1 1 3 3 2 2]
5	[3 2; 4 0; 5 1; 2 4; 0 3; 1 1; 0 0]	[3 3 3 2 2 1 1]
6	[2 3; 1 4; 0 1; 1 1; 2 0; 4 2; 5 3]	[2 2 1 1 1 3 3]
7	[2 1; 1 0; 1 3; 2 4; 0 5; 4 2; 5 0]	[1 1 1 2 2 3 3]
8	[4 0; 5 1; 3 1; 2 0; 1 1; 2 4; 0 3]	[3 3 3 1 1 2 2]
9	[1 3; 2 4; 0 5; 4 1; 3 3; 0 0; 1 1]	[2 2 2 3 3 1 1]
10	[0 1; 1 0; 3 2; 4 0; 5 2; 1 3; 2 4]	[1 1 3 3 3 2 2]

Список литературы

1. Медведев В. С. Нейронные сети / В. С. Медведев, В. Г. Потемкин. – М.: Диалог МИФИ, 2002.

2. Дьяконов В. П. MATLAB 5.3.1 с пакетами расширений / В. П. Дьяконов, И. В. Абраменкова, В. В. Круглов. – М: Нолидж,2001.

3. Комашинский В. И. Нейронные сети и их применение в системах управления и связи / В. И. Комашинский, Д. А. Смирнов. – М.: Горячая линия – Телеком, 2002.