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
«__1__» _____ 2018 г.

МЯГКИЕ ВЫЧИСЛЕНИЯ И НЕЙРОННЫЕ СЕТИ

Методические рекомендации по организации и выполнению практических занятий для обучающихся по программам высшего образования по направлениям 12.06.01 и 09.06.01

Курск 2018

УДК 004.93:61

Составитель: С.А. Филист.

Рецензент
Доктор технических наук, профессор Р.А. Томакова

Мягкие вычисления и нейронные сети: методические рекомендации по организации и выполнению практических занятий/ Юго-Зап. гос. ун-т; сост.: С.А. Филист. - Курск, 2018. - 38 с.

Предназначены для обучающихся по программам высшего образования по направлениям 12.06.01 «Фотоника, приборостроение, оптические и биотехнические системы и технологии (Приборы, системы и изделия медицинского назначения)» и 09.06.01 «Информатика и вычислительная техника (Системный анализ, управление и обработка информации (технические и медицинские системы))».

Текст печатается в авторской редакции

Подписано в печать 1.03.18. Формат 60x84 1/16.
Усл.печ.л. 1,7. Уч.-изд.л. 1,6 Тираж 100 экз. Заказ: 1456 Бесплатно. Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Практическая 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.

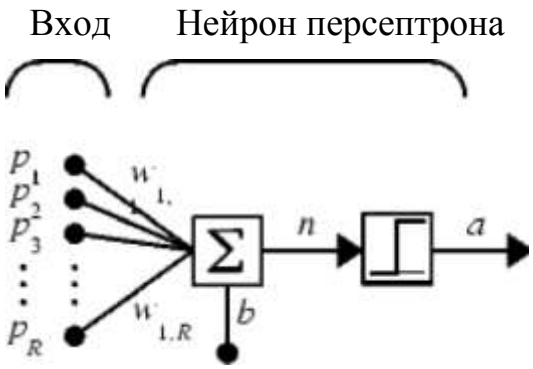
Теоретические сведения

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 (рисунок 1.1).



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

Рисунок 1.1 - Нейрон, используемый в модели персептрона

Каждое значение элемента вектора входа персептрона умножено на соответствующий вес w1j,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 рисунке 1.2, для персептрона с двумя входами и смещением.

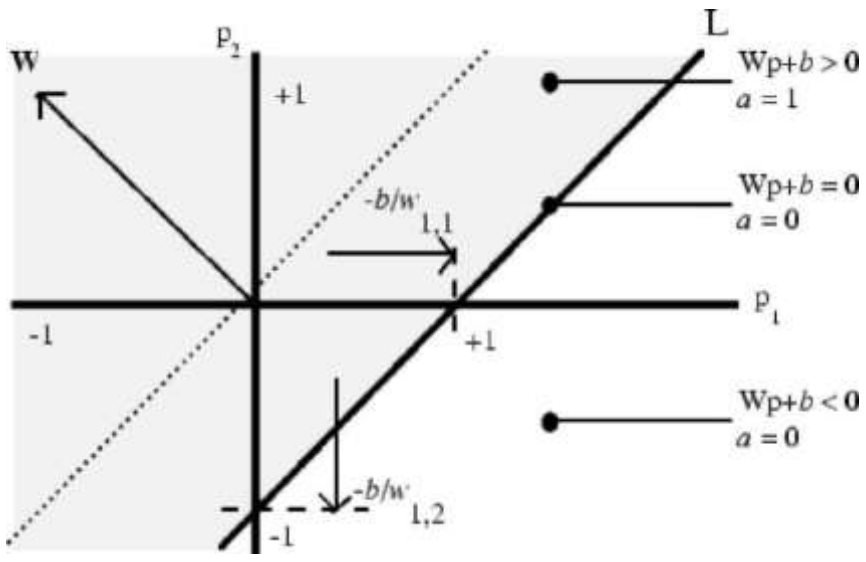


Рисунок 1.2 - Персептрон с двумя входами и смещением

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

$$W^T p + b = 0, \qquad (1.1)$$

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 рисунке 1.2.

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

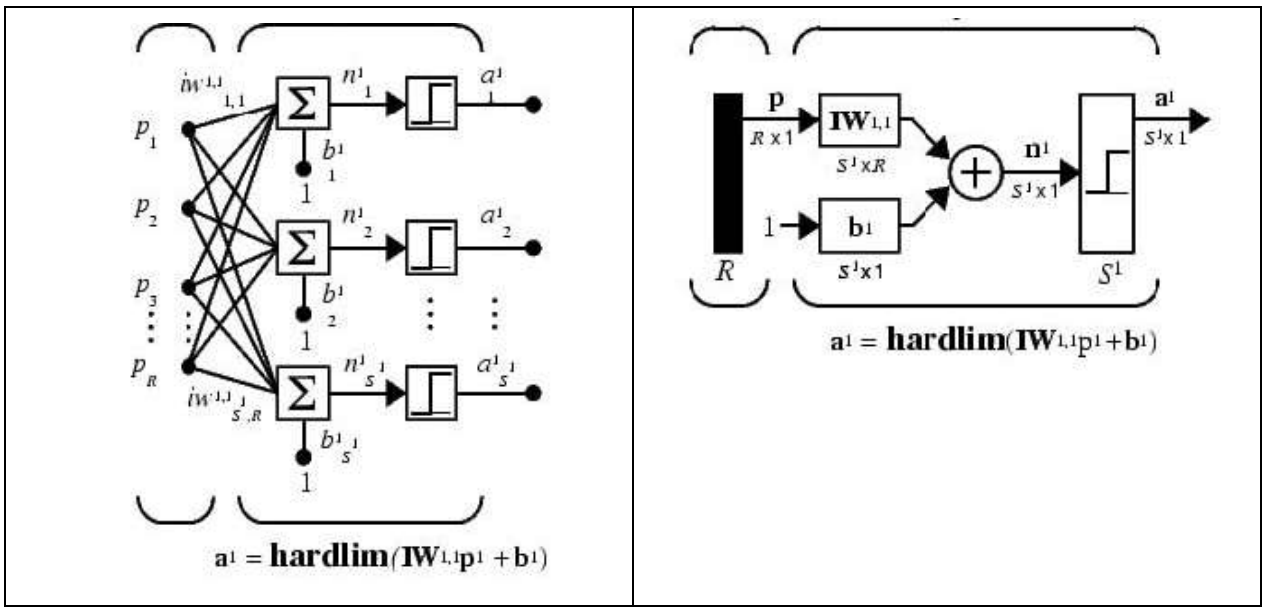
Архитектура сети

Персептрон состоит из единственного слоя, включающего *S* нейронов, как это показано на рисунке 1.3,*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

$$a = f \cdot (W_p + b),, \qquad (1.2)$$

Входы Слой нейронов Вход Слой 1



а б

Рисунок 1.3 - Персептрон состоит из единственного слоя, включающего *S* нейронов

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. 1.2.

Определим реакцию сети на входные векторы p_1 и p_2,расположенные по разные стороны от разделяющей линии:

```
pl = [1; 1];
al = sim(net,p1 % Моделирование сети net с входным
вектором pl
```

```
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.

Технология выполнения

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

Практическая работа №2
«Процедуры настройки параметров персептронных нейронных сетей. Правила настройки. Процедура адаптации»

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

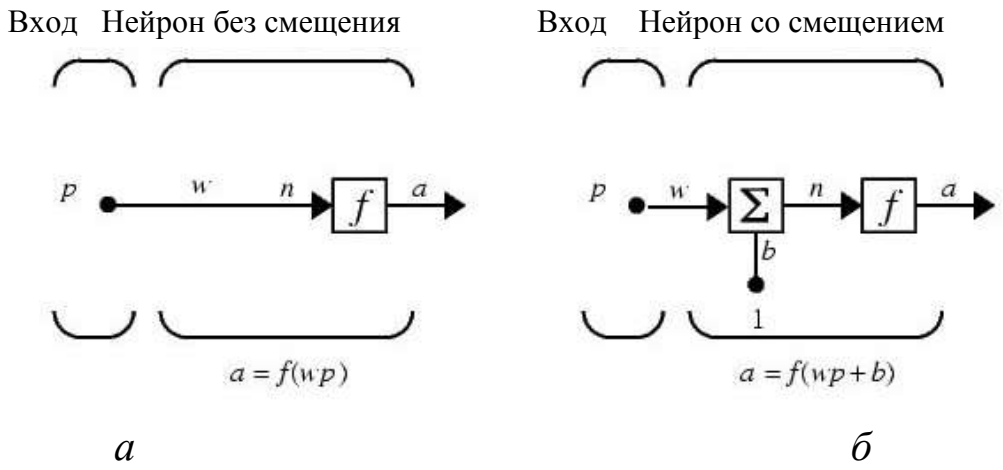
Теоретические сведения

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



а *б*

Рисунок 2.1 – Нейрон

Уравнение нейрона со смещением имеет вид

$$a = f(w * p + b * 1). \qquad (2.1)$$

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

$$n = [w\ b]\begin{bmatrix} p \\ 1 \end{bmatrix} = w^{*}p + b^{*}1. \qquad (2.2)$$

Нейрон с векторным входом

Нейрон с одним вектором входа *p* с *R* элементами p_1, p_2,..., p_R показан на рисунке 2.2. 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

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

или

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

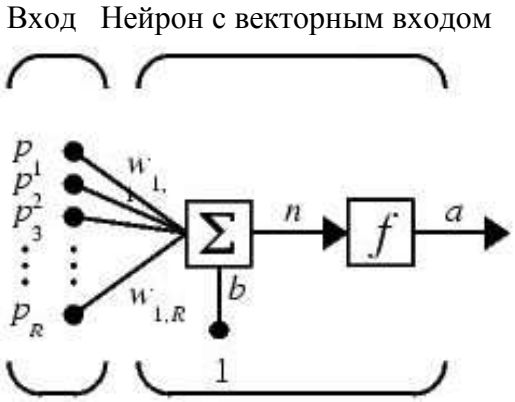


Рисунок 2.2 – Нейрон с одним вектором

и 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 (рисунок 2.3).

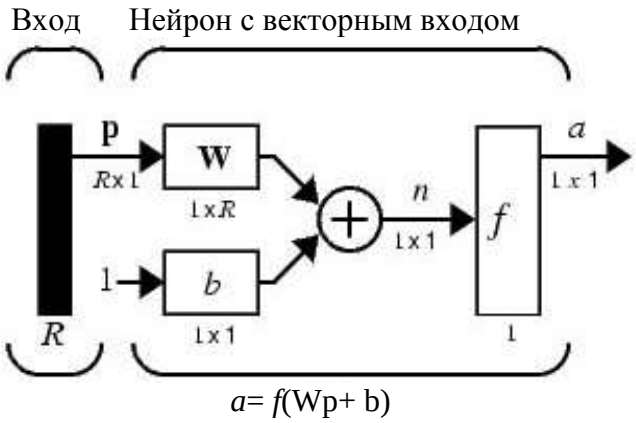


Рисунок 2.3- Нейрон с векторным входом

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*×l.

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 (рисунок 2.3). Эта система обозначений поясняет строение сети и связанную с ней матричную математику.

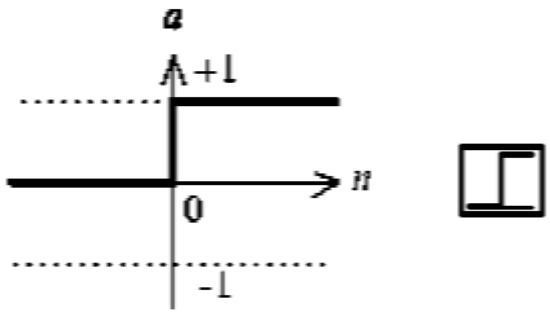
Функции активации

Функции активации (передаточные функции) нейрона могут иметь самый различный вид. Функция активации f как правило, принадлежит к классу сигмоидальных функций, которые имеют

две горизонтальные асимптоты и одну точку перегиба, с аргументом функции n(входом) и значением функции (выходом) *a*.

Рассмотрим три наиболее распространенные формы функции активации.

Единичная функция активации с жестким ограничением hardlim Эта функция описывается соотношением *a*= hardlim(n) = = 1(n) и показана на рисунке 2.4.



a = *hardlim*(*n*)

Рисунок 2.4 – Функция a = hardlim(n)

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 рисунке 2.5.

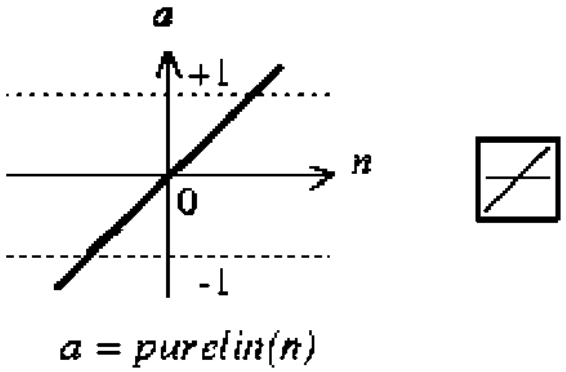


Рисунок 2.5 – Функция *a*= purelin(n)

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 $a = \text{logsig}(n) = 1/(1 + \exp(-n))$ и показана на рисунке 2.6.

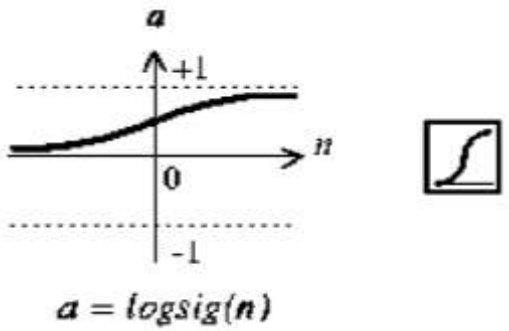


Рисунок 2.6 – Функция *a*=logsig(n)

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 рисунке 2.7, где *а* – ступенчатая, *б*– линейная, *в* – логистическая.

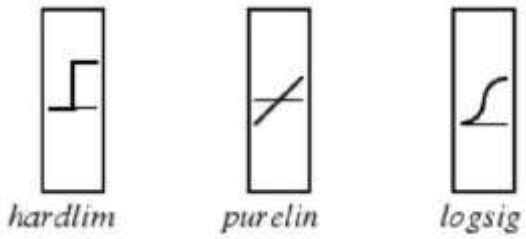


Рисунок 2.7 – Функции

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 (рисунок 2.8):

```
x= –5:0.1:5;
plot(x,sin(x))
```

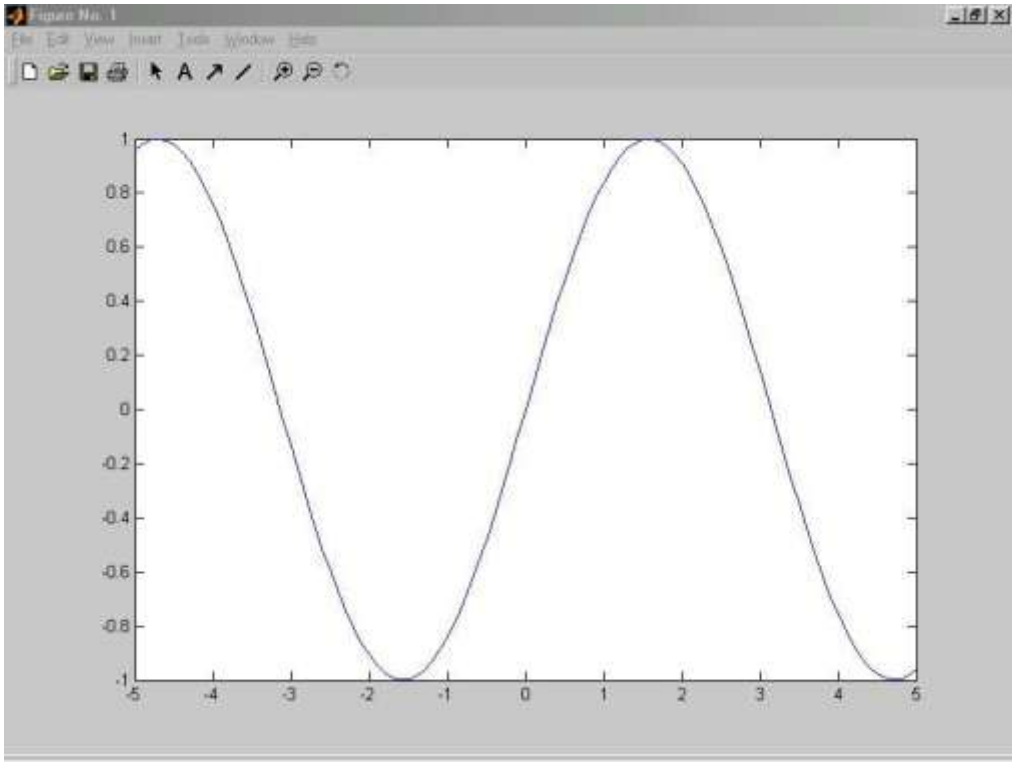


Рисунок 2.8 – Оператор plot

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.

Технология выполнения

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

Практическая 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

$$\{p_1, t_1\}, \{p_2, t_2\},...,\{p_Q, t_Q\}. \qquad (3.1)$$

Требуется минимизировать одну из следующих функций квадратичной ошибки:

$$mse = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q} (t(k) - a(k))^2 \qquad (3.2)$$

Или

$$sse = \sum_{k=1}^{Q} e(k)^2 = \sum_{k=1}^{Q} (t(k) - a(k))^2, \qquad (3.3)$$

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 рисунке 3.1.

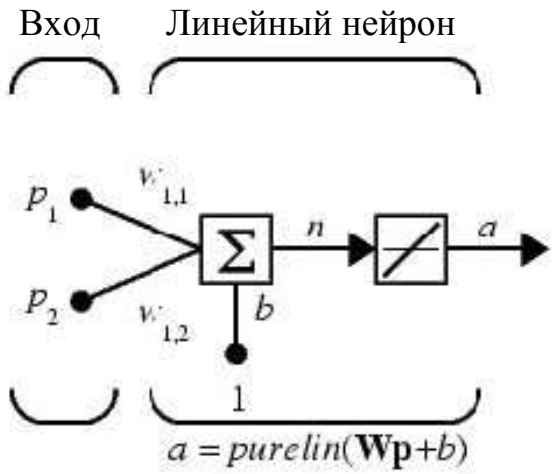


Рисунок 3.1 - Структурная схема этого линейного нейрона

Запишем уравнение выхода нейрона

$$a = \text{purelin}(n) = \text{purelin}(wp + b) = wp + b. \quad (3.4)$$

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 рисунке 3.2.

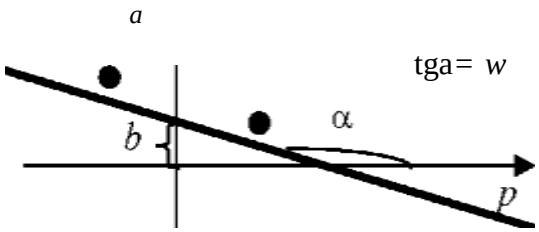


Рисунок 3.2 - Графическая интерпретация настройки веса и смещения

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 (рисунок 3.3):

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

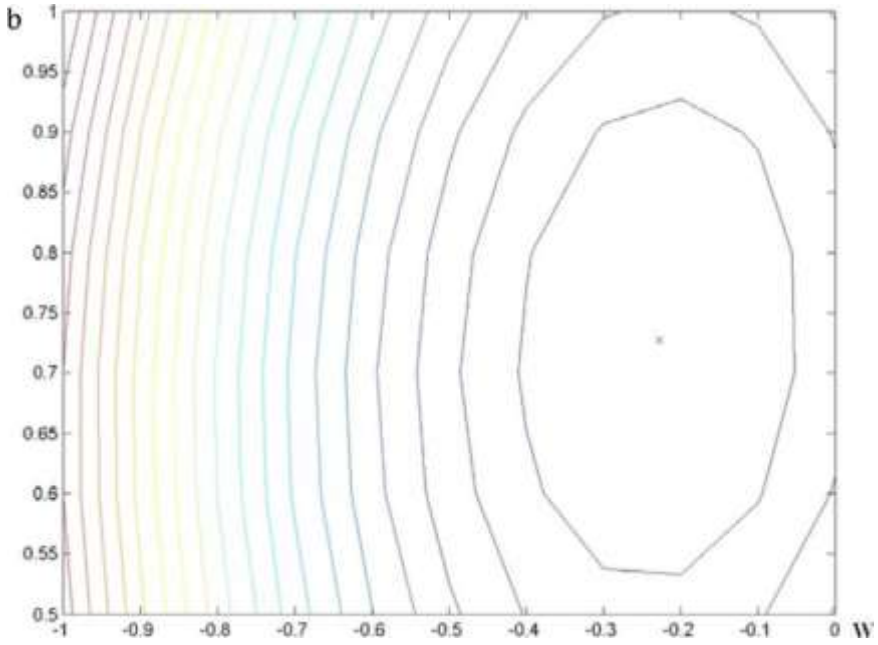


Рисунок 3.3 – Критерии качества

Технология выполнения

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}

Практическая 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.

Теоретические 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 P^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 $LW^{2,1}$ и смещений 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

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

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
```

Технология выполнения

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}

Практическая работа № 5

«Сети PNN»

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

Теоретические сведения

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

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 рисунке 5.1.

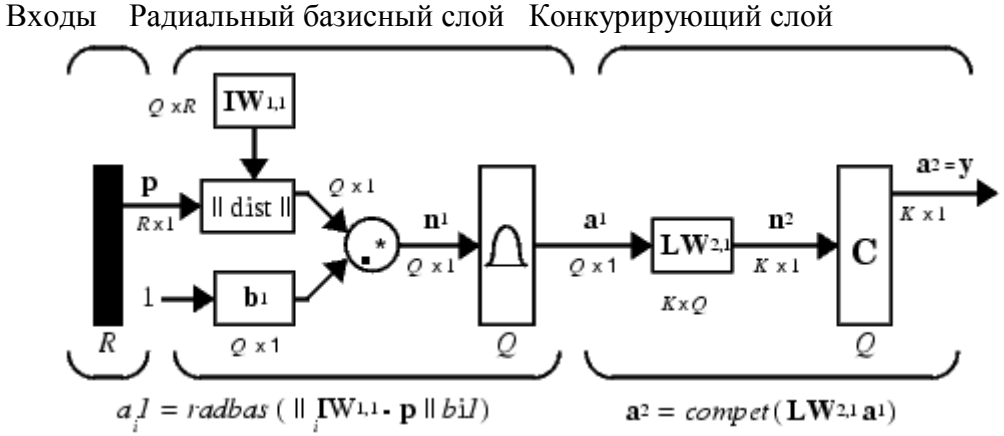


Рисунок 5.1 – Структура сети PNN

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 $K \times Q$, 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 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 a^1 числом близким к 1.

Весовая матрица второго слоя 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 $T^* a^1$ определяет элементы вектора $\mathbf{a}^1$, соответствующие каждому из K 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.

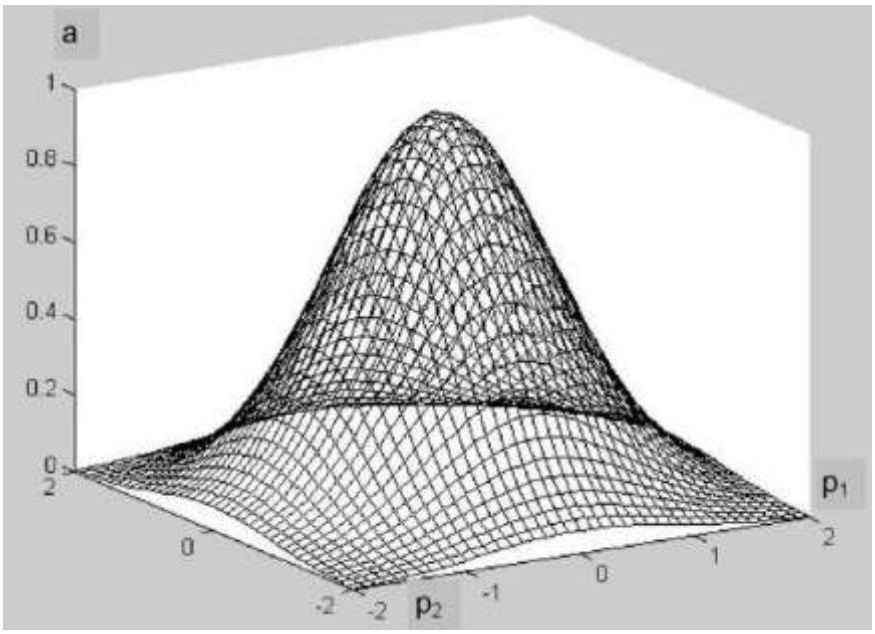


Рис. 2

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 нейроны второй 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:

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

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

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
```

Технология выполнения

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]