Нейронные сети с MATLAB.
3 курс, 6 семестр
доцент кафедры ДУиСА Голубева Л.Л.

Лабораторная работа
"Исследование персептронных нейронных сетей"

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

Теоретические сведения

Персептроном называется простейшая нейронная сеть, веса и смещения которой могут быть настроены таким образом, чтобы решить задачу классификации входных векторов.

Модель нейрона

Нейрон, используемый в модели персептрона, имеет ступенчатую функцию активацию **`hard-lim`** с жестким ограничением (рис.1 б). Нейрон персептрона возвращает 1, если вход функции активации $n \geq 0$, и 0, если $n<0$. Результирующая сумма n равна

$$n=w_{1,1}p_1+w_{1,2}p_2+\ldots + w_{1,R}p_R + b$$

и служит аргументом функции активации **`hardlim`**.

Для иллюстрации представлен однонейронный персептрон с одним двухэлементным вектором входа (рис.1 а).

а) б)

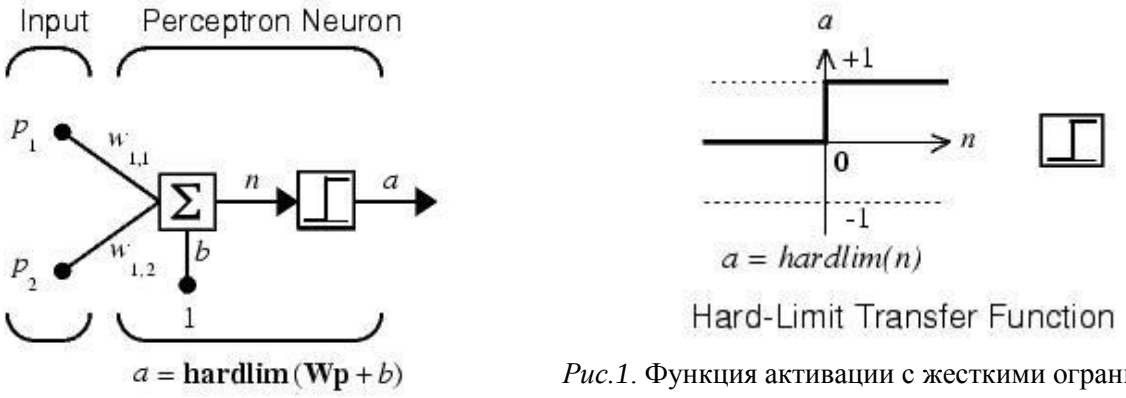


Рис.1. Функция активации с жесткими ограничениями

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

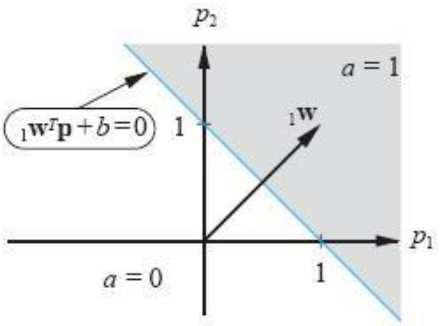


Рис.2. Деление персептроном пространства входов на 2 области

Пространство входов делится на 2 области разделяющей линией ***L*** (*decision boundary line* ***L***), которая для двумерного случая задается уравнением

$$\mathbf{w}^{\mathrm{T}}\mathbf{p} + b = 0.$$

Линия ***L*** 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.

Архитектура персептронной сети

Персептрон – это однослойная нейронная сеть с S нейронами и R входами, каждый из которых может состоять из Q элементов. Передаточной функцией каждого нейрона является ступенчатая функция типа **`hardlim`** или **`hardlims`**. Элементы входов и смещения взвешиваются с помощью функции скалярного произведения **`dotprod`** и суммируются с помощью функции накопления **`netsum`**.

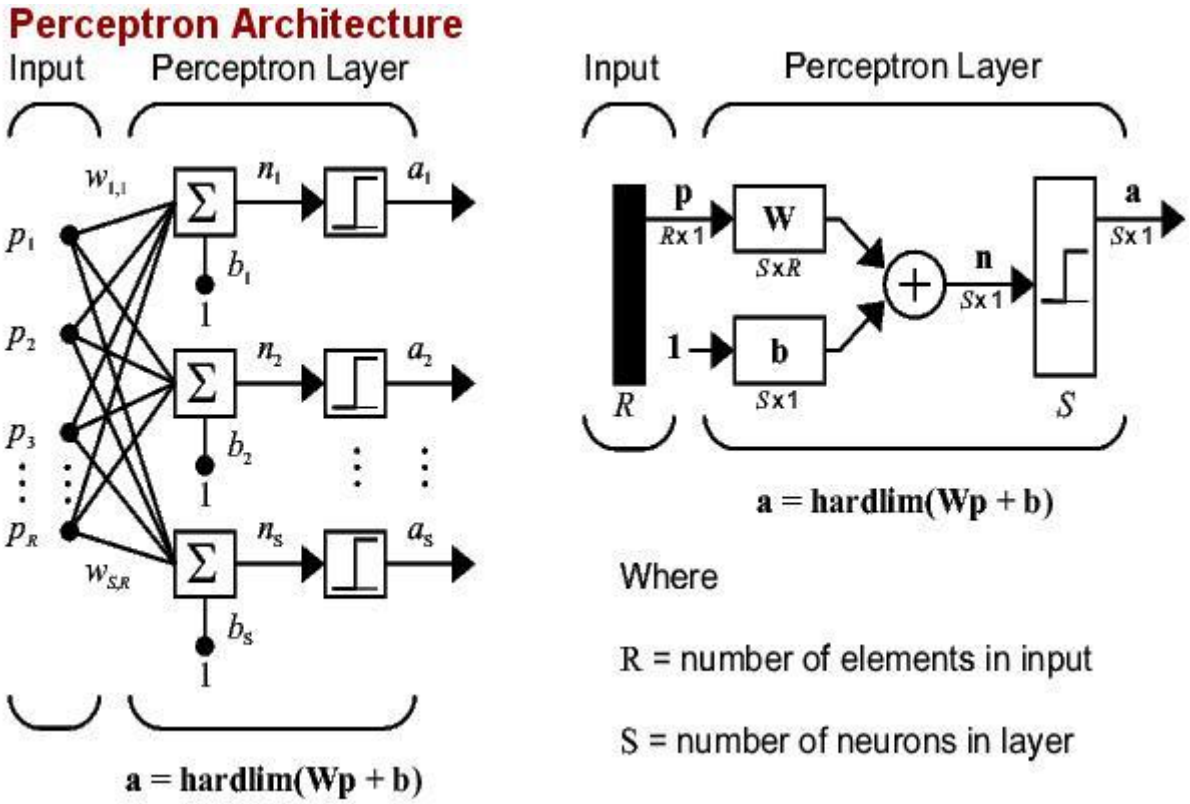


Рис.3. Архитектура персептронной сети

Линейная разделимость

Линейная разделимость ограничивает однослойные сети задачами классификации, в которых множества точек (соответствующих входным значениям) могут быть разделены геометрически. Для случая **с двумя входами** разделитель является *прямой линией*. В случае **трех входов** разделение осуществляется *плоскостью, рассекающей трехмерное пространство*.

Преодоление ограничения линейной разделимости

Серьезное ограничение представляемости однослойными сетями проблемы *линейной разделимости* можно преодолеть, **добавив дополнительные слои**. Например, **двухслойные сети** можно получить каскадным соединением двух однослойных сетей. Они способны выполнять более общие классификации, отделяя те точки, которые содержатся в выпуклых ограниченных или неограниченных областях (рис.4).

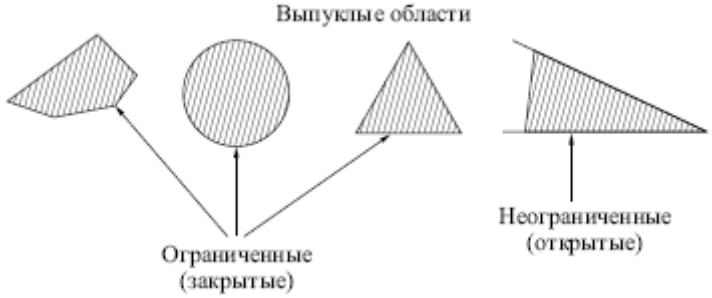


Рис.4. Выпуклые области

Двухслойные сети. На рис.5 а) каждый нейрон слоя 1 разбивает плоскость XOY на две полуплоскости, один обеспечивает единичный выход для входов ниже верхней линии, другой – для входов выше нижней линии. Выходной нейрон второго слоя реализует логическую функцию **И.** На рис.5 б) показан результат такого двойного разбиения, где выходной сигнал нейрона второго слоя равен единице только внутри V-образной области.

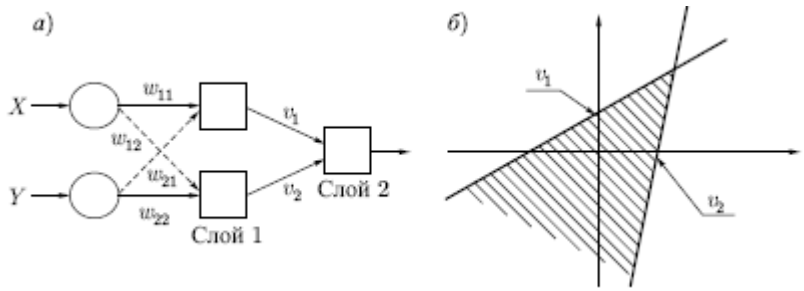


Рис.5. Двухслойный перцептрон

Во входном слое может быть использовано **три нейрона** с дальнейшим разбиением плоскости и созданием области треугольной формы. Включением достаточного числа нейронов во входной слой может быть образован **выпуклый многоугольник** любой желаемой формы. Все такие многогранники **выпуклы**, так как они образованы с помощью операции **И** над областями, задаваемыми линиями: следовательно, *только выпуклые области* и возникают. Точки, не составляющие выпуклой области, не могут быть отделены от других точек плоскости двухслойной сетью.

Нейрон второго слоя не ограничен функцией **И**. Он может реализовывать многие другие функции при подходящем выборе весов и порога.

Входы не обязательно должны быть двоичными. **Вектор непрерывных входов** может представлять собой произвольную точку на плоскости XOY. В этом случае сеть способна *разбивать плоскость на непрерывные области*.

Трехслойная сеть есть более общий случай. Ограничения на выпуклость отсутствуют. Нейрон третьего слоя принимает в качестве входа набор выпуклых многоугольников, и их логическая комбинация может быть *невыпуклой* (рис.6 б).

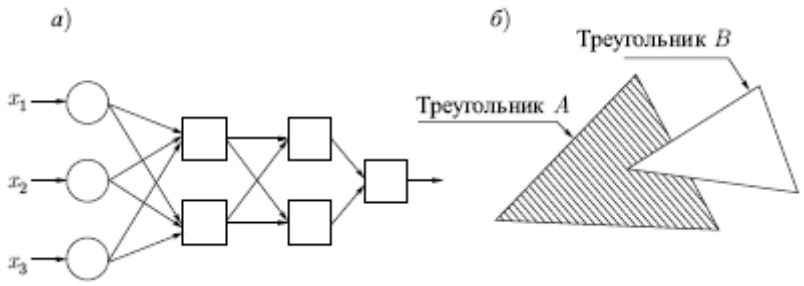


Рис.6. Трехслойный перцептрон для распознавания невыпуклых областей

Моделирование персептрона средствами MATLAB

Для формирования модели однослойного персептрона в системе MATLAB предназначена функция **newp**, которая может быть вызвана 2 способами. Первый вызов:

net = newp(PR,S,TF,LF), где:

PR – *R*×2 матрица минимальных и максимальных значений для *R* входных элементов;

S – число нейронов;

TF – функция активации **hardlim** или **hardlims** (default = 'hardlim')

LF – обучающая функция **learnp** или **learnpn** (default = 'learnp')

и возвращает новый персептрон **net**.

Либо вызов:

net = newp(P,T), где:

P – матрица R×Q из Q входных векторов по R элементов каждый;

T – матрица S×Q из Q целевых векторов по S элементов каждый

и возвращает новый персептрон **net**.

В новых версиях MATLAB используется функция **perceptron**:

perceptron(hardlimitTF, perceptronLF)

hardlimitTF – передаточная функция (по умолчанию hardlim),

perceptronLF – обучающее правило (по умолчанию learnp).

Примеры

Пример 1. Функция

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

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

Пример 2. Функция

```
P = [0 0 1 1; 0 1 0 1];
T = [0 1 1 1];
net = newp (P,T)
```

создает персептрон с одним двухэлементным вектором входа и одним нейроном.

Пример 3. Функция

```
P = [0 0 1 1; 0 1 0 1];
T = [0 1 1 1];
net = peceptron;
net=train(net,P,T);
view(net)
Y=net(P);
```

решает задачу логического ИЛИ.

Пример 4. Рассмотрим тестовый пример для развития понимания, как должно работать обучающее правило персептрона (**p** – вектор входа, **t** – целевой вектор , **a** – выход сети).

$$e = t - a.$$
$$_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}.$$
$$b^{new} = b^{old} + e. \qquad (1)$$

Постановка задачи (**Test Problem**). Пары `input/targets` для рассматриваемой тестовой задачи имеют вид:

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

Графическая интерпретация задачи представлена на рис.7 a), где два входных вектора $\mathbf{p}_2$ и $\mathbf{p}_3$, имеющих цель 0, представлены незакрашенными кружками, а входной вектор $\mathbf{p}_1$ с целью 1 – закрашенным кружком.

Сеть для решения данной задачи представляет собой персептрон с двухэлементным входом, одним нейроном, одним выходом, без смещения. Сеть имеет только два параметра $w_{1,1}$ и $w_{1,2}$.

Концепция обучающих правил (**Constructing Learning Rules**). Обучение начинается с задания некоторых начальных значений параметров сети. В данном случае мы обучаем 2-input/1-output сеть без смещения, поэтому необходимо проинициализировать только 2 весовых коэффициента $w_{1,1}$ и $w_{1,2}$. Элементы вектора весовых коэффициентов мы инициализируем случайными значениями, рис.7 b).

$$_1\mathbf{w}^T = [1.0\ -0.8].$$

Теперь подаем на вход сети входные векторы, начиная с вектора $\mathbf{p}_1$.

$$a = hard\lim\left({}_1w^T p_1\right) = hard\lim\left(\begin{bmatrix}1.0 & -0.8\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}\right) = hard\lim(-0.6) = 0.$$

Для вектора $\mathbf{p}_1$ имеем случай 2, т.е. выход a=0, а цель t_1=1, и входной вектор $\mathbf{p}_1$ класса 1 неверно классифицирован как класс 0. Начальному вектору весов $_1\mathbf{w}$ соответствует разделяющая линия L , которая неправильно классифицирует вектор $\mathbf{p}_1$. Необходимо изменить вектор весов, так чтобы он указывал больше в направлении вектора $\mathbf{p}_1$, чтобы в будущем иметь лучший шанс классифицировать его правильно.

Применим правило (1) к нашей задаче для корректировки вектора весов $_1\mathbf{w}$. Согласно формуле (1) для случая 2 необходимо выполнить сложение векторов $_1\mathbf{w}$ и $\mathbf{p}_1$, как показано на рис.7 c).

$$ {}_1w^{new} = {}_1w^{old} + p_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}. $$

Следующим подаем на вход в сеть входной вектор $\mathbf{p}_2$. Имеем:

$$ a = hard\lim\left({}_1w^T p_2\right) = hard\lim\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) = hard\lim(0.4) = 1. $$

Для вектора $\mathbf{p}_2$ имеем случай 3, т.е. выход a=1, а цель t_2=0, и входной вектор $\mathbf{p}_2$ класса 0 неверно классифицирован как класс 1. Теперь необходимо сдвинуть вектор весов от направления вектора входа $\mathbf{p}_2$. Согласно формуле (1) для случая 3 необходимо выполнить вычитание векторов ${}_1\mathbf{w}$ и $\mathbf{p}_2$, как показано на рис.7 d).

$$ {}_1w^{new} = {}_1w^{old} - p_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}. $$

Следующим подаем на вход в сеть входной вектор $\mathbf{p}_3$. Имеем:

$$ a = hard\lim\left({}_1w^T p_3\right) = hard\lim\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) = hard\lim(0.8) = 1. $$

Для вектора $\mathbf{p}_3$ имеем случай 3, т.е. выход a=1, а цель t_3=0, и входной вектор $\mathbf{p}_3$ класса 0 неверно классифицирован как класс 1. Теперь необходимо сдвинуть вектор весов от направления вектора входа $\mathbf{p}_3$. Согласно формуле (1) для случая 3 необходимо выполнить вычитание векторов ${}_1\mathbf{w}$ и $\mathbf{p}_3$, как показано на рис.7 e).

$$ {}_1w^{new} = {}_1w^{old} - p_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}. $$

Рис.7 e) показывает, что персептрон, наконец, обучен классифицировать все три входных ветра правильно. Так, для вектора $\mathbf{p}_1$ имеем:

$$ a = hard\lim\left({}_1w^T p_1\right) = hard\lim\left(\begin{bmatrix} 3.0 & 0.2 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) = hard\lim(3.4) = 1. $$

Для вектора $\mathbf{p}_2$ имеем:

$$ a = hard\lim\left({}_1w^T p_2\right) = hard\lim\left(\begin{bmatrix} 3.0 & 0.2 \end{bmatrix}\begin{bmatrix} -1 \\ 2 \end{bmatrix}\right) = hard\lim(-2.6) = 0. $$

Для вектора $\mathbf{p}_3$ имеем:

$$ a = hard\lim\left({}_1w^T p_3\right) = hard\lim\left(\begin{bmatrix} 3.0 & 0.2 \end{bmatrix}\begin{bmatrix} 0 \\ -1 \end{bmatrix}\right) = hard\lim(-0.2) = 0. $$

Согласно формуле (1) корректировку весов проводить не нужно. Весовые параметры настроены правильно, выходы равны целям для всех трех векторов входа, процесс обучения персептрона успешно завершен.

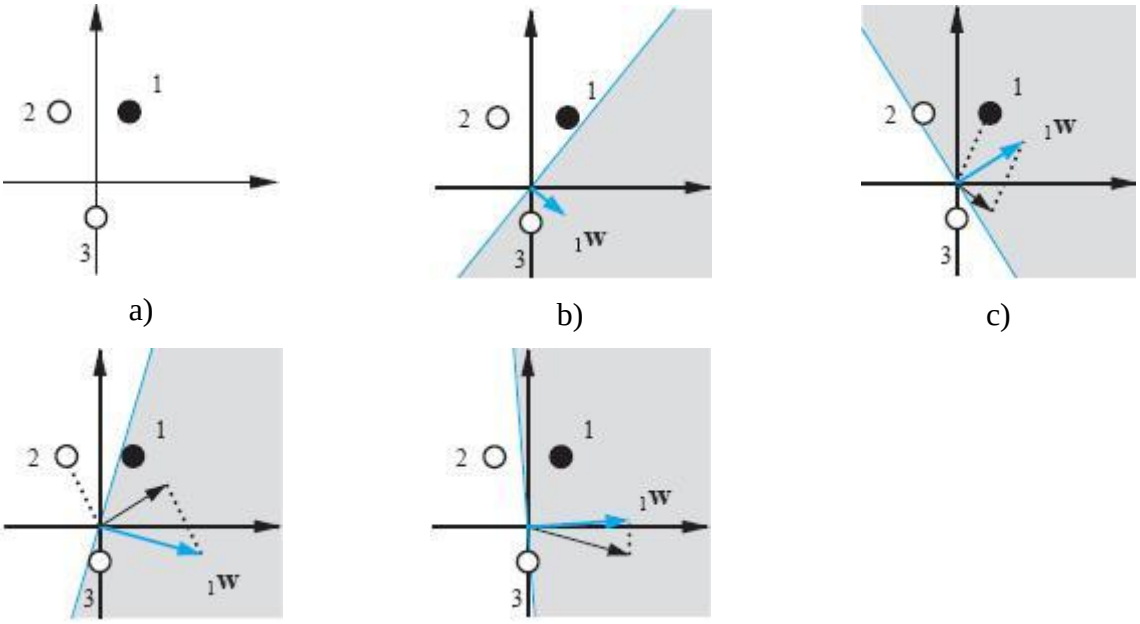


Рис.7. Графическое изображение алгоритма обучения персептрона

Представленный ниже код подтверждает правильность выполненных расчетов.

```
% learning rule for perceptron
net=newp([-2 2; -2 2],1);    % create perceptron
net.b{1}=[0];                % no bias
w=[1 -0.8];
net.IW{1,1}=w;               % set random weights
% input vector p1, class 1
p1=[1;2];                    % input
t=[1];                       % target
a=sim(net,p1)                % simulation and output
a =
     0
e=t-a                        % error
e =
     1
dw=learnp(w,p1,[],[],[],[],e,[],[],[]) % learning rule
dw =
     1     2
w=w+dw                       % corrected weights
w =
    2.0000    1.2000
net.IW{1,1}=w;               % set corrected weights
% input vector p2, class 0
p2=[-1;2];                   % input
t=[0];                       % target
a=sim(net,p2)                % simulation and output
a =
     1
e=t-a                        % error
```

```
e =
    -1
dw=learnp(w,p2,[],[],[],[],e,[],[],[]) % learning rule
dw =
     1    -2
w=w+dw                          % corrected weights
w =
    3.0000   -0.8000
net.IW{1,1}=w;                  % set corrected weights
% input vector p3, class 0
p3=[0;-1];                      % input
t=[0];                          % target
a=sim(net,p3)                   % simulation and output
a =
     1
e=t-a                           % error
e =
    -1
dw=learnp(w,p3,[],[],[],[],e,[],[],[]) % learning rule
dw =
     0     1
w=w+dw                          % corrected weights
w =
    3.0000    0.2000
net.IW{1,1}=w;                  % set corrected weights
% test
a=sim(net,p1)
a =
     1
a=sim(net,p2)
a =
     0
a=sim(net,p3)
a =
     0
```

Процедура адаптации. Для настройки (обучения) персептрона применяется функция адаптации **adapt**, которая корректирует параметры персептрона по результатам каждого входного вектора. Можно ввести поочередно каждый входной вектор и, установив параметр **passes** (число проходов) равным 1, выполнить по одному проходу для каждого входного вектора.

```
net.adaptParam.passes = 1;
net = adapt(net,p1,t1);
net = adapt(net,p2,t2);
net = adapt(net,p3,t3);
```

Но можно выполнить эту процедуру автоматически, задав сразу все обучающее множество в виде массива ячеек и выполнив 1 проход:

```
net=newp([-2 2; -2 2],1);    % create perceptron
% input vectors
P={[1;2] [-1;2] [0; 1]}      % inputs
```

```
P =
    [2x1 double]    [2x1 double]    [2x1 double]
T={[1] [0] [0]}              % targets
T =
    [1]    [0]    [0]
net.adaptParam.passes = 1;
net = adapt(net,P,T);       % pass 1
a=sim(net,P)
a =
    [0]    [0]    [0]
net = adapt(net,P,T);       % pass 2
a=sim(net,P)
a =
    [0]    [0]    [0]
net = adapt(net,P,T);       % pass 3
a=sim(net,P)
a =
    [1]    [0]    [0]
```

или задав сразу все обучающее множество в виде матрицы:

```
net=newp([-2 2; -2 2],1);    % create perceptron
% input vectors
P=[[1;2] [-1;2] [0; 1]]      % inputs
P =
     1    -1     0
     2     2     1
T=[[1] [0] [0]]              % targets
T =
     1     0     0
net.adaptParam.passes = 1;
net = adapt(net,P,T);       % pass 1
a=sim(net,P)
a =
     0     0     0
net = adapt(net,P,T);       % pass 2
a=sim(net,P)
a =
     0     0     0
net = adapt(net,P,T);       % pass 3
a=sim(net,P)
a =
     1     0     1
net = adapt(net,P,T);       % pass 4
a=sim(net,P)
a =
     1     0     0
e=T-a                        % error
e =
     0     0     0
```

Если рассчитанные выходы персептрона не совпадают с целевыми значениями, то необходимо выполнить еще несколько циклов настройки, вызывая функцию **adapt** и проверяя правильность полученных результатов. Вместо этого можно сразу увеличить число проходов, задав, например, параметр **net.adaptParam.passes = 10.**

```
net.adaptParam.passes = 10;
net = adapt(net,P,T);
a=sim(net,P)                % simulation and output
a =
     1     0     0
e=T-a                       % error
e =
     0     0     0
```

Практические задания

Задание 1. Функции к применению. Изучите работу функций **newp**, **perceptron**, **ginput**, **gscatter**, **plotpv**, **plotpc**.

Задание 2. Изучите работу персептронных сетей на демонстрационных моделях пакета MATLAB (*Help\Demos\Toolboxes\Neural Network\Perceptrons*):

- **nnd4db** – Decision boundaries demonstration.
- **nnd4pr** – Perceptron rule demonstration.

Задание 3. Создать персептрон с одним нейроном и одним двухэлементным вектором входа, значения элементов которого изменяются в диапазоне от **–2** до **2**, настроить веса и смещение для реализации разделяющей линии

$$-p_1 + p_2 - 1 = 0,$$

а затем с помощью моделирования определить классы значений входного вектора, выполнив следующие действия:

1. Создать персептрон:

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

2. Произвести ручную инициализацию:

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

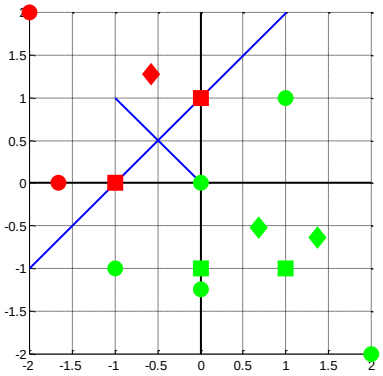
3. Произвести проверку работы персептрона:

```
p = [1;1];
a = sim(net, p)     % a = 0;
p = [-1;1];
a = sim(net, p)     % a = 1.
```

4. Определить классы значений вектора:

```
p1=[[-2;-2] [-2;-1] [-2;0] [-2;1] [-2;2] [-1;-2] [-1;-1] [-1;0]…
    [-1;1] [-1;2] [0;-2] [0;-1] [0;0] [0;1] [0;2] [1;-2]…
    [1;-1]   [1;0]  [1;1] [1;2] [2;-2] [2;-1] [2;0] [2;1] [2;2]];
a1 = sim(net, p1)            % [0]-0-й класс;  [1]-1-й класс.
```

5. Выполнить графическую визуализацию работы персептрона.



Задание 4. Используя **GUI NNTool** (команда **>> nntool**) подготовить обучающее множество **P=[[0;0] [0;1] [1;0] [1;1]]** и **T=[0 1 1 1]**, создать персептрон с одним нейроном и одним двухэлементным бинарным входом и обучить его выполнению логической функции **OR**. Экспортировать обучающее множество и персептрон в рабочее пространство **Workspace** и выполнить графическую визуализацию работы персептрона.

Задание 5. Создать персептрон с одним нейроном и одним двухэлементным бинарным входом и обучить его выполнению логической функции **AND**, выполнив следующие действия:

1. Создать персептрон:

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

2. Подготовить обучающие последовательности:

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

или:

```
P={[0;0] [0;1] [1;0] [1;1]};
T={0 0 0 1};
```

3. Настроить параметры персептрона для выполнения логической функции **AND**, используя ***процедуру адаптации***:

```
net.adaptParam.passes = 10;     % – число проходов;
net.adapt(net,p,T1);            % – настройка на AND;
net.IW{1,1},net.b{1}            % – разделительная линия w1*p1 +w2*p2+b=0;
Y =sim(net,p)                   % – моделирование AND.
```

Либо настроить параметры персептрона для выполнения логической функции **AND**, используя ***процедуру обучения***:

```
net.trainParam.epochs=20;       % – число циклов;
net = init(net);                % – инициализация;
net = train(net,p,T2);          % – настройка на AND;
net.IW{1,1},net.b{1}            % – линия w1*p1 +w2*p2+b=0;
Y = sim(net,p)                  % – моделирование AND.
```

4. Выполнить графическую визуализацию работы персептрона.

Задание 6. Используя пошаговый режим адаптации и обучения, проследить изменения весов, смещения, выходного значения и ошибки для персептрона, рассмотренного в Задании 3. Для этих целей использовать команды:

```
net.adaptParam.passes = 1;
[net,Y,e]=adapt(net,p,T);          % – для AND;
net.trainParam.epochs = 1;
[net,Y,e]=train(net,p,T);          % – для OR;
net.IW{1,1},net.b{1}               % – настраиваемые параметры.
```

Задание 7. Создать, обучить и апробировать персептрон для принятия решения о зачислении в высшее учебное заведение абитуриентов, сдавших вступительные экзамены. Количество экзаменов задается (Preliminary Examination Number). Каждый экзамен оценивается по N-балльной шкале и имеет заданную нижнюю границу (Lower Limit). Зачислению подлежат абитуриенты, преодолевшие заданный суммарный порог (Total Threshold). Например,

1) Два экзамена: Preliminary Examination Number = 2;
   10-бальная шкала: N = 10;
   каждая оценка не ниже 3 баллов: Lower Limit = 3;
   проходной балл составляет: Total Threshold = 11.

Задание 8. Для заданного преподавателем варианта (таблица):

1) выполнить ***ручной расчет*** настройки весов и смещений персептронной нейронной сети;
2) разработать и реализовать в MATLAB алгоритм создания и моделирования персептронной нейронной сети;
3) определить параметры созданной нейронной сети (веса и смещение) и проверить правильность работы сети для последовательности входных векторов (не менее 5);
4) сравнить результаты ручных расчетов и расчетов, выполненных в системе MATLAB.

Номер варианта	Количество входов – 2; количество нейронов – 1		
	Диапазоны значений входов	Входы персептрона	Целевые выходы
1	–4…+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	–4…+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	–4…+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}

Задание 9. В интерактивном режиме задать две 2D-точки (функция **`ginput`**) в области [-2 2; -2 2] плоскости XOY. Данные точки определяют разделяющую линию L плоскости.

Для персептрона с одним двухэлементным входом **p** подобрать весовые коэффициенты и смещение таким образом, чтобы персептрон выполнял классификацию входных векторов с разделением на две области. Вводить точки входа интерактивно и выполнять графическую визуализацию.

Задание 10. При помощи персептрона решить и визуализировать задачу распознавания заданного интерактивно (функция **ginput**):

а) треугольника;

б) выпуклого четырехугольника.

Задание 11. При помощи персептрона решить и визуализировать задачу распознавания невыпуклых областей, изображенную на рис.6.

Задание 12. При помощи персептрона решить и визуализировать задачу классификации, представленную на рис.8.

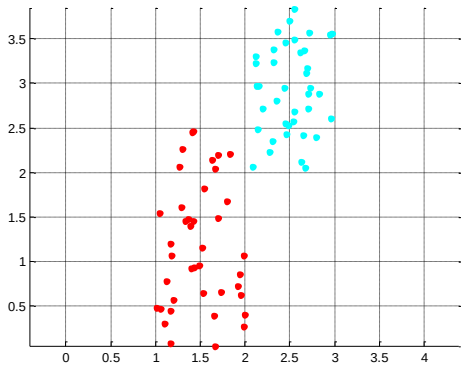


Рис.8. Задача классификации

Задание 13.

Персептрон с трех элементным входом и одним нейроном с заданными параметрами

```
net=newp([-2 2; -2 2; -2 2],1); % create perceptron
net.IW{1,1}=[2 3 -1];
net.b{1}=-4;          % 2x+3y-z-4=0
```

образует разделяющую плоскость ***L***: $2p_1+3p_2-p_3-4=0$, которая делит пространство входов $\mathbb{R}^3$ на два полупространства (рис.9). Разделяющая плоскость ***L*** перпендикулярна к вектору весов **w** и смещена на величину *b*.

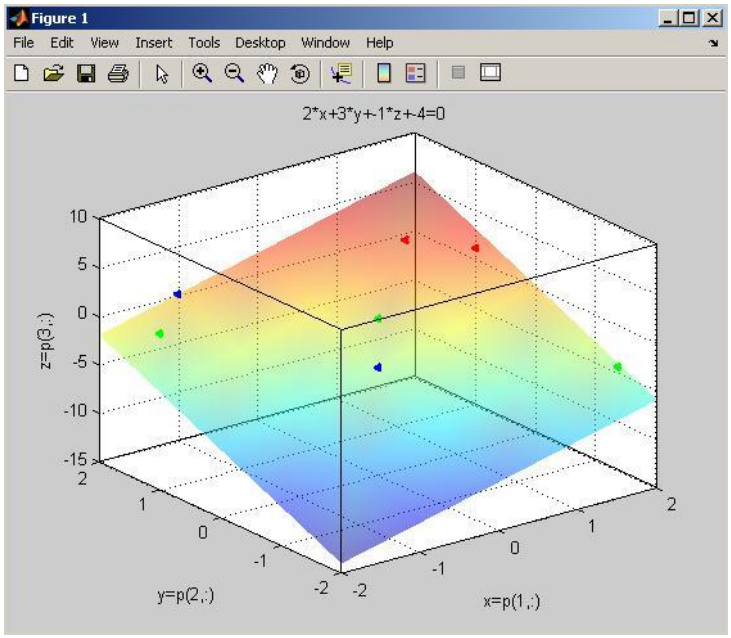


Рис.9. Разделяющая плоскость

Задайте три точки, определяющие разделяющую плоскость L. Рассчитайте вектор весов персептрона, перпендикулярный плоскости L. Задайте 10 случайных точек в рассматриваемом пространстве и визуализируйте работу персептрона.

Задание 14. Решите при помощи персептрона задачу классификации:

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

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

Задание 15. Решите при помощи персептрона задачу классификации:

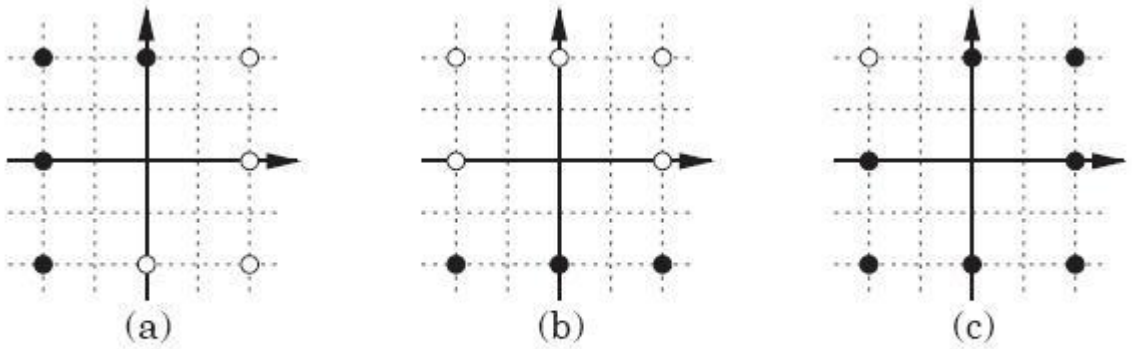


Figure P4.1 Simple Classification Problems

Задание 16. Задача классификации с 4 классами входных векторов. Решите при помощи однослойного двухнейронного персептрона задачу классификации.

Векторы входа обучающего множества:

$$\text{class 1:}\left\{\mathbf{p}_1=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{p}_2=\begin{bmatrix}1\\2\end{bmatrix}\right\},\ \text{class 2:}\ \left\{\mathbf{p}_3=\begin{bmatrix}2\\-1\end{bmatrix},\mathbf{p}_4=\begin{bmatrix}2\\0\end{bmatrix}\right\},$$

$$\text{class 3:}\left\{\mathbf{p}_5=\begin{bmatrix}-1\\2\end{bmatrix},\mathbf{p}_6=\begin{bmatrix}-2\\1\end{bmatrix}\right\},\ \text{class 4:}\ \left\{\mathbf{p}_7=\begin{bmatrix}-1\\-1\end{bmatrix},\mathbf{p}_8=\begin{bmatrix}-2\\-2\end{bmatrix}\right\}.$$

Целевые векторы обучающего множества:

$$\text{class 1:}\left\{\mathbf{t}_1=\begin{bmatrix}0\\0\end{bmatrix},\mathbf{t}_2=\begin{bmatrix}0\\0\end{bmatrix}\right\},\ \text{class 2:}\ \left\{\mathbf{t}_3=\begin{bmatrix}0\\1\end{bmatrix},\mathbf{t}_4=\begin{bmatrix}0\\1\end{bmatrix}\right\},$$

$$\text{class 3:}\left\{\mathbf{t}_5=\begin{bmatrix}1\\0\end{bmatrix},\mathbf{t}_6=\begin{bmatrix}1\\0\end{bmatrix}\right\},\ \text{class 4:}\ \left\{\mathbf{t}_7=\begin{bmatrix}1\\1\end{bmatrix},\mathbf{t}_8=\begin{bmatrix}1\\1\end{bmatrix}\right\}.$$

Персептрон для решения данной задачи имеет вид (рис.10):

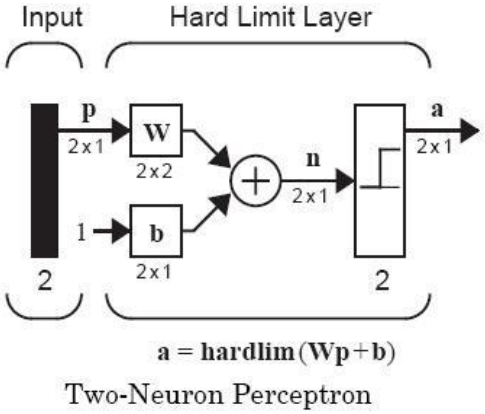


Рис.10. Двухнейронный однослойный персептрон

Обучающее множество отображено красным цветом (рис.11а), результаты выполнения классификации персептроном после обучения – синим (рис.11б). Точки, подлежащие классификации, вводятся интерактивно (функция **`ginput`**)

а)	б)

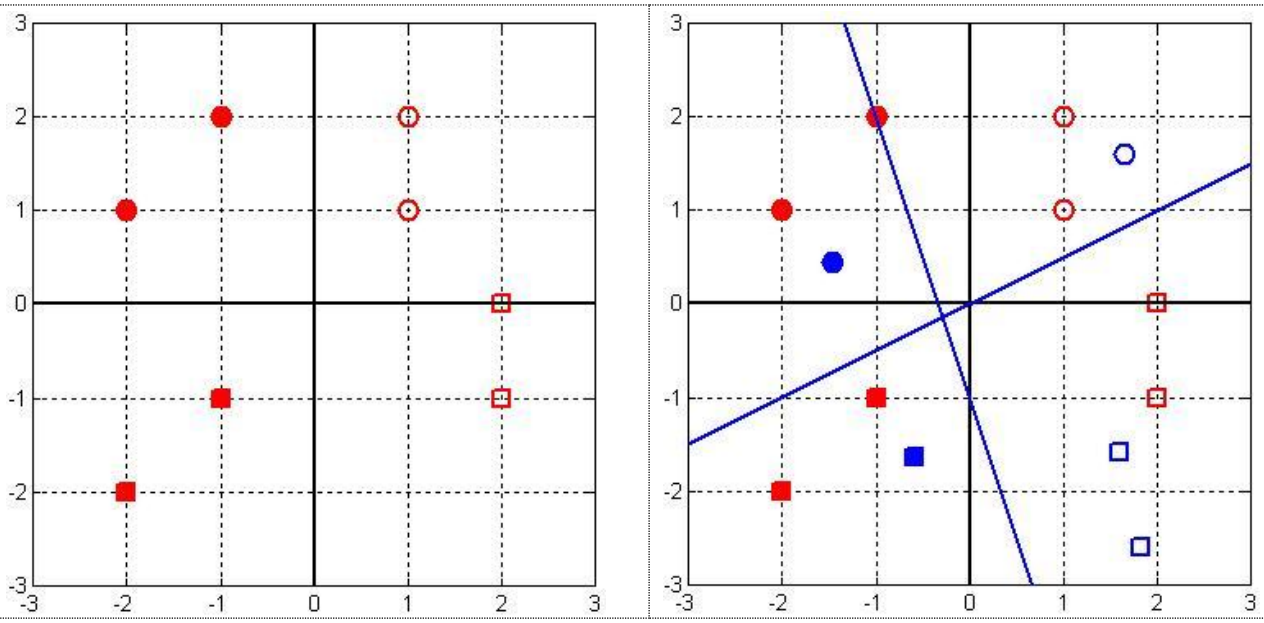


Рис.11. Решение персептроном задачи классификации с 4 классами входных векторов

Литература

1. H.Demuth, M.Beale. Neural Network Toolbox. For use with MATLAB. 840 p. (MathWorks_nnet.pdf, part 3. *Perceptrons*)
2. M.Beale, M. Hagan, H.Demuth. Neural Network Toolbox™. User's Guide (MathWorks_nnet_ug.pdf, parts: "*Historical Networks. Perceptron Networks*" on page 9-3)
3. Медведев В. С. Нейронные сети. MATLAB 6. Учебно-справочное издание / В. С. Медведев, В. Г. Потемкин. – М.: Диалог МИФИ, 2002 (*разд. 4. Персептроны*).