**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа №1**

**по курсу «Нейроинформатика»**

**Персептроны. Процедура обучения Розенблатта**

Выполнил: Дубинин А. О.

Группа: 80-407Б-17

Преподаватель: Аносова Н.П.

Москва 2021.

Вариант 11.

Выполнение работы:

* 1. Обучающее множество занести в отчет.

P = [-3.9 4.5 0.8 2.5 0 3.9;  
 -0.1 -1.6 -2.8 -2.5 1.9 4.5];  
T = [0 1 0 0 0 1];

1.2 Создать сеть. Сконфигурировать сеть под обучающее множество. Отобразить структуру сети с помощью функции display и результат занести в отчет.

net = newp([-5 5; -5 5], [0 1]); % построили сеть  
display(net); % информация о сети в консоль  
view(net); % сеть на экран

Neural Network

name: 'Perceptron'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 1

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 3

sampleTime: 1

connections:

biasConnect: true

inputConnect: true

layerConnect: false

outputConnect: true

subobjects:

input: Equivalent to inputs{1}

output: Equivalent to outputs{1}

inputs: {1x1 cell array of 1 input}

layers: {1x1 cell array of 1 layer}

outputs: {1x1 cell array of 1 output}

biases: {1x1 cell array of 1 bias}

inputWeights: {1x1 cell array of 1 weight}

layerWeights: {1x1 cell array of 0 weights}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: 'dividetrain'

divideParam: (none)

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mae'

performParam: .regularization, .normalization

plotFcns: {'plotperform', plottrainstate,

plotconfusion}

plotParams: {1x3 cell array of 3 params}

trainFcn: 'trainc'

trainParam: .showWindow, .showCommandLine, .show, .epochs,

.time, .goal, .max\_fail

weight and bias values:

IW: {1x1 cell} containing 1 input weight matrix

LW: {1x1 cell} containing 0 layer weight matrices

b: {1x1 cell} containing 1 bias vector

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

evaluate: outputs = net(inputs)

1.3 Реализовать алгоритм обучения по правилу Розенблатта. Код алгоритма занести в отчет.

*% 1.3***function** net = Rosenblatt(net, P, T, iters, learnRate)  
 **for** j = 1:iters  
 **for** i = 1:6  
 p = P(:,i);  
 t = T(:,i);  
 a = sim(net, p);  
 e = t - a;   
 **if** (mae(e))  
 net.IW{1,1} = net.IW{1,1} + p' \* e \* learnRate;  
 net.b{1} = net.b{1} + 1 \* e \* learnRate;  
 **end  
 end** disp([**'Итерация: '**, num2str(j)]);  
 disp(**'Ошибка: '**);  
 disp(mae(T - net(P)));  
 disp(**'Веса: '**);  
 disp(net.IW{1,1});  
 disp(**'Смещения: '**);  
 disp(net.b{1});  
 **end  
end**

1.3.1 Инициализировать сеть случайными значениями. Для инициализированных весов и смещений использовать функцию rands. Занести в отчет весовые коэффициенты и смещения.

*% инициализировали сеть случайными значениями*net.inputWeights{1,1}.initFcn = **'rands'**; *% весовые коэффициенты*net.biases{1}.initFcn = **'rands'**; *% смещения*net = init(net);

1.3.1

IW =

0.5155 0.4863

b =

0.3575

1.3.2 Рассчитать два цикла обучения сети по правилу. Для расчета выходов сети использовать функцию *net*. В качестве показателя качества обучения использовать функцию *mae*. Занести в отчет весовые коэффициенты и смещения после расчета каждой эпохи (итерации). Также занести в отчет ошибку обучения сети по всей обучающей выборке (*mae(T - net(P))*).

net = Rosenblatt(net, P, T, 2, 0.3);

Итерация: 1

Ошибка:

0.1667

Веса:

0.2375 0.3031

Смещения:

-1.5311

Итерация: 2

Ошибка:

0.1667

Веса:

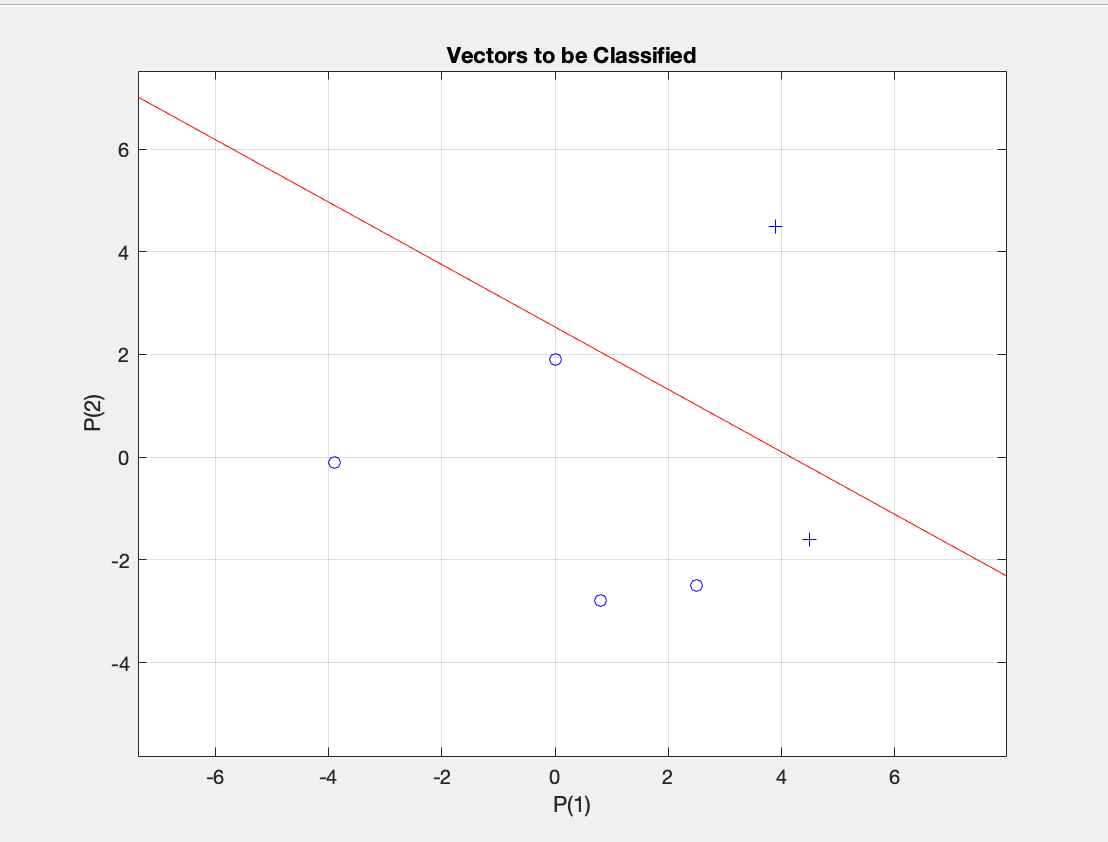
0.5975 0.8431

Смещения:

-2.1311

1.3.3 После обучения отобразить обучающую выборку и дискриминантную линию. Для отображения использовать функции plotpv и plotpc соответственно. Также отобразить сетку с помощью функции grid.

*%1.3.3*line2 = plotpc(net.IW{1}, net.b{1}); *% рисуем прямую*



1.4 Провести обучение сети с помощью встроенной функции train и проверить качество обучения. Занести в отчет окно Neural Network Training.

1.4.1 Инициализировать сеть случайными значениями.

*%1.4.1*net = init(net);  
disp(**'1.4.1: '**);  
disp(**'IW: '**);  
disp(net.IW{1,1});  
disp(**'b: '**);  
disp(net.b{1});

1.4.1:

IW:

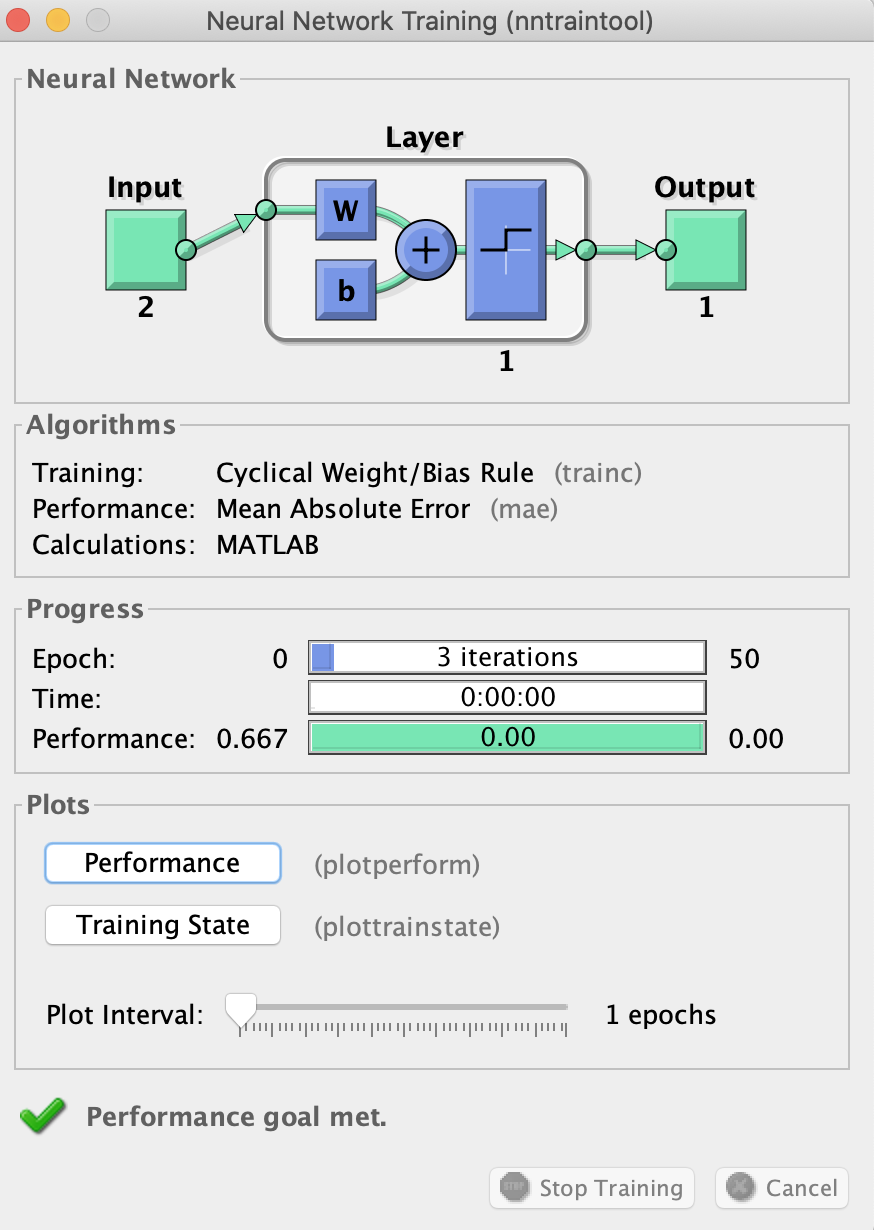
-0.6756 0.5886

b:

-0.3258

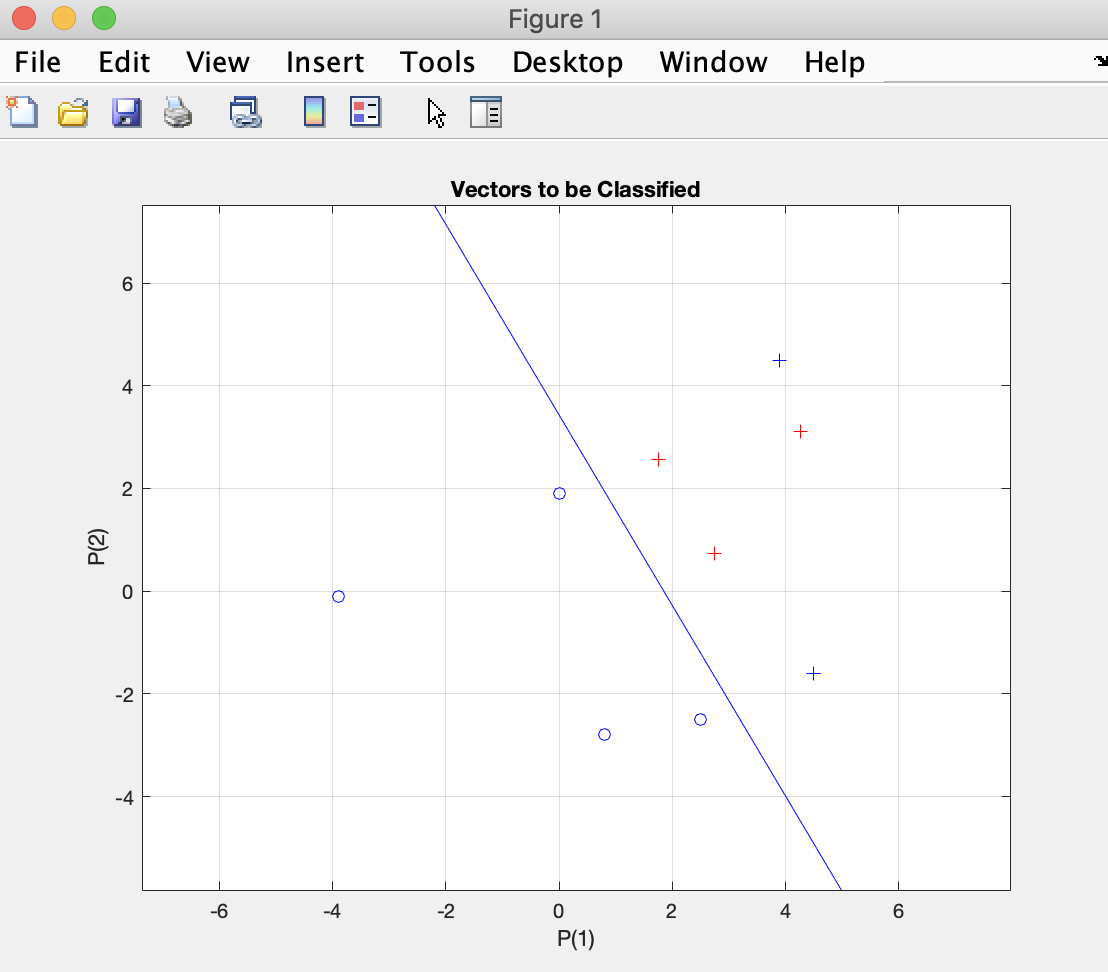
1.4.2 Провести обучение сети с помощью функции train c числом эпох равным 50. Если необходимо, то произвести обучение несколько раз. Занести в отчет весовые коэффициенты и смещения.

*%1.4.2  
% Выполним 50 итераций с помощью функции train*net.trainParam.epochs = 50;  
net = train(net, P, T);  
  
disp(**'После 50 итераций:'**)  
disp(**'Веса : '**);  
disp(net.IW{1,1});  
disp(**'Смещения: '**);  
disp(net.b{1});



1.4.3 Проверить качество обучения: случайным образом задать 3 точки и классифицировать их. Для генерации случайных чисел использовать функцию rands. Отобразить сетку, дополнительные точки, обучающую выборку, и дискриминантную линию. Результаты занести в отчет.

*%1.4.3  
% создаем набор тестовых точек*testpoints = 5 \* rand(2,3);  
testclasses = net(testpoints);  
  
*% рисуем их*plotpv(testpoints, testclasses);  
point = findobj(gca,**'type'**,**'line'**);  
set(point,**'Color'**,**'red'**);  
hold on  
plotpv(P, T)  
plotpc(net.IW{1},net.b{1});  
grid on  
hold off



2. Изменить обучающее множество так, чтобы классы стали линейно неразделимыми. Проверить возможности обучения по правилу Розенблатта.

2.1 Изменить обучающее множество.

P = [-3.9 4.5 0.8 2.5 0 3.9 0;  
 -0.1 -1.6 -2.8 -2.5 1.9 4.5 0];  
T = [0 1 0 0 0 1 1];

2.2 Инициализировать сеть случайными значениями.

*% 2.2*net.inputWeights{1,1}.initFcn = **'rands'**;  
net.biases{1}.initFcn = **'rands'**;  
net = init(net);  
disp(**'2.2: '**);  
disp(**'IW: '**);  
disp(net.IW{1,1});  
disp(**'b: '**);  
disp(net.b{1});

2.2:

IW:

0.8588 0.5514

b:

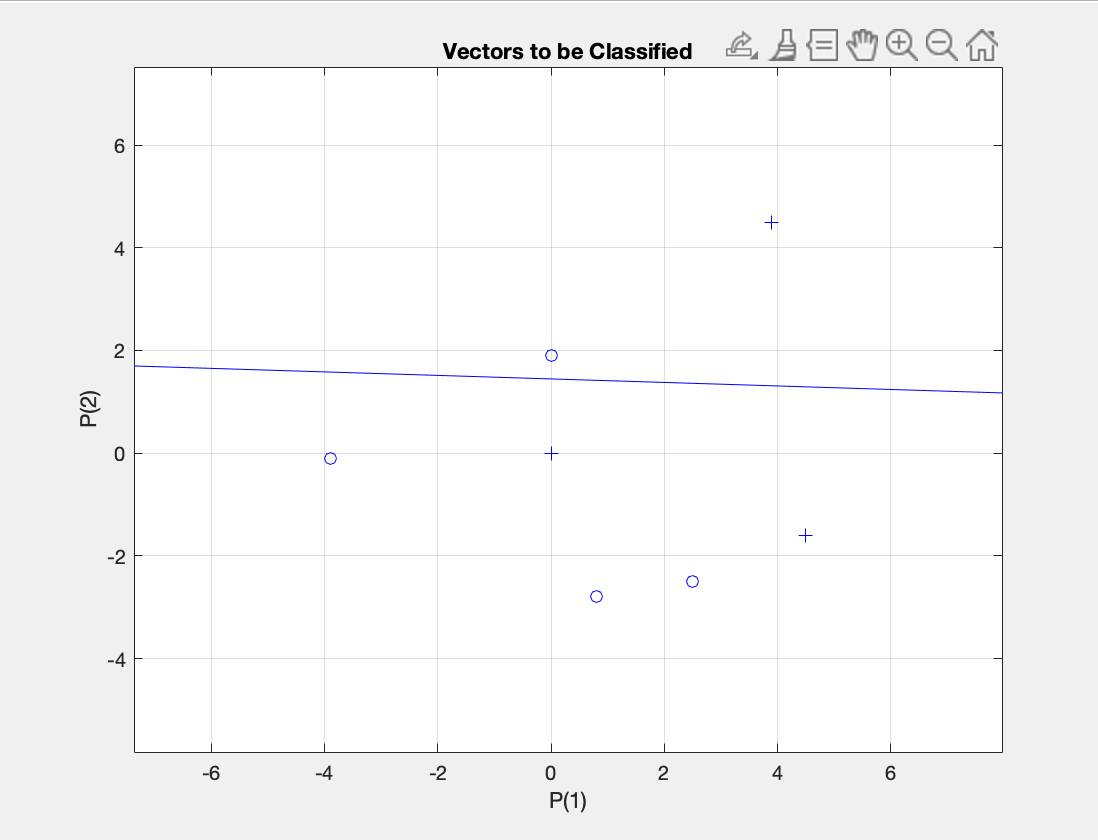
-0.8377

2.3 Провести обучение сети с помощью функции train c числом эпох равным 50. Отобразить обучающую выборку и полученную дискриминантную линию. Результаты занести в отчет.

MAE:

ans =

0.2857



3. Для второй обучающей выборки построить и обучить сеть, которая будет правильно относить точки к четырем классам.

3.1 Обучающее множество занести в отчет.

P =[ 3.9 -4.6 2.7 -3.3 -2.9 4 -4 -4.5;  
 -4.1 0.5 -1.9 -1.7 0.1 1.2 -1.1 0 ];  
T = [0 1 0 0 1 1 0 1;  
 1 0 1 0 0 1 0 0];

3.2 Создать сеть.

*% 3.2*net = newp([-5 5; -5 5], 2);  
view(net);

3.3 Инициализировать сеть случайными значениями.

*% 3.3*net.inputWeights{1,1}.initFcn = **'rands'**;  
net.biases{1}.initFcn = **'rands'**;  
net = init(net);  
disp(**'3.3: '**);  
disp(**'IW: '**);  
disp(net.IW{1,1});  
disp(**'b: '**);  
disp(net.b{1});

3.4 Провести обучение сети с помощью функции train c числом эпох равным 50. Если необходимо, то произвести обучение несколько раз. Занести в отчет весовые коэффициенты и смещения. Занести в отчет окно Neural Network Training.

*% 3.4*net.trainParam.epochs = 50;  
net = train(net, P, T);  
disp(**'Веса: '**);  
disp(net.IW{1,1});  
disp(**'Смещения : '**);  
disp(net.b{1});  
  
plotpv(P, T);  
plotpc(net.IW{1}, net.b{1});  
grid;

Веса:

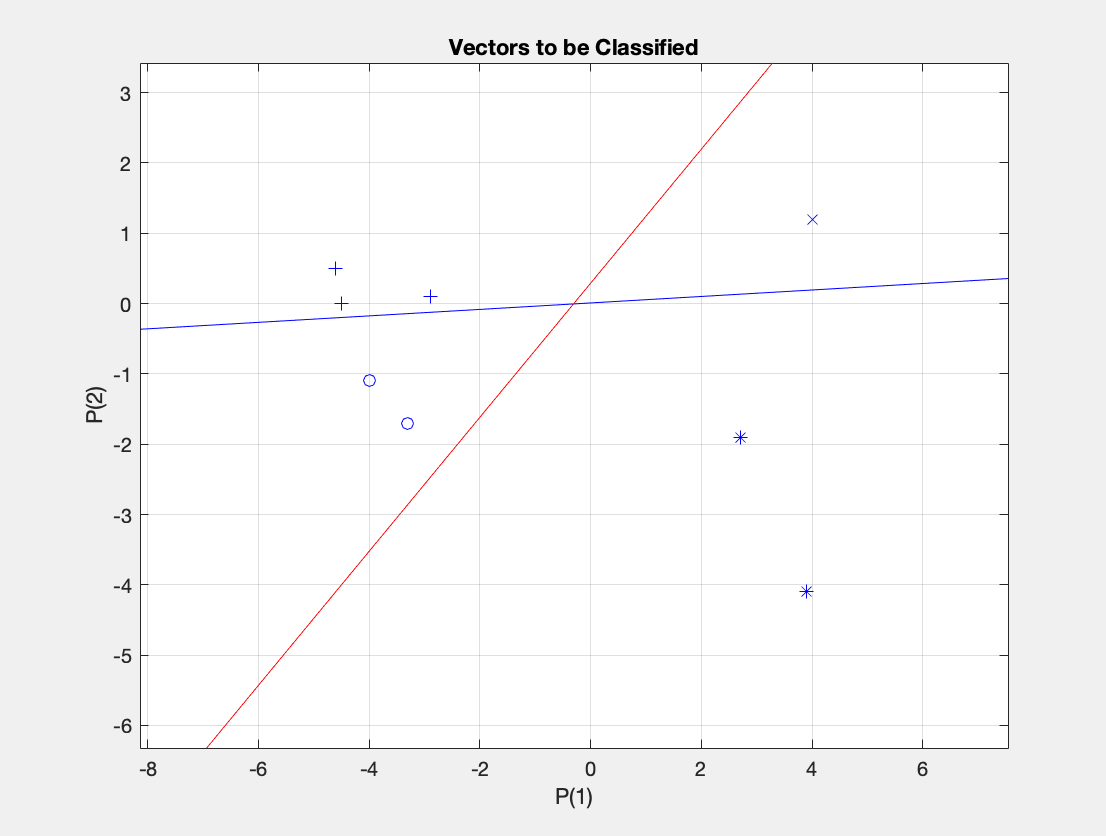
-0.1254 2.7262

3.8177 -4.0064

Смещения:

-0.0228

1.1571



3.5 Проверить качество обучения: случайным образом задать 5 точек и классифицировать их. Отобразить сетку, дополнительные точки, обучающую выборку, и дискриминантную линию. Результаты занести в отчет.

*% 3.5*testpoints = 5\*rand(2,5);  
testclasses = net(testpoints);  
plotpv([P testpoints], [T, testclasses]);   
point = findobj(gca,**'type'**,**'line'**);  
set(point,**'Color'**,**'red'**);  
hold on  
plotpv(P, T);  
plotpc(net.IW{1}, net.b{1});  
hold off

