Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 1

по курсу «Нейроинформатика»

Тема: Персептроны. Процедура обучения Розенблатта.

Студент: Ваньков Д. А.
Группа: 80-407Б-17
Преподаватель: Аносова Н.П.

Москва, 2021

Постановка задачи

Цель работы: Исследование свойств персептрона Розенблатта и его применение для решения задачи распознавания образов.

1. Для первой обучающей выборки построить и обучить сеть, которая будет правиль- но относить точки к двум классам. Отобразить дискриминантную линию и проверить качество обучения.

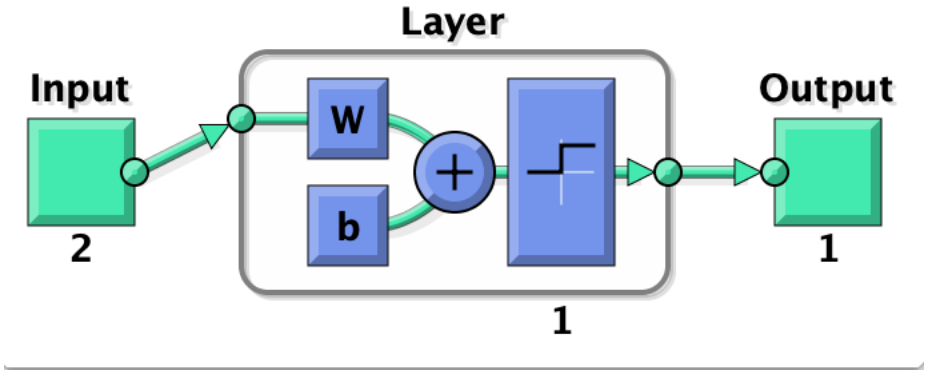
2. Изменить обучающее множество так, чтобы классы стали линейно неразделимы- ми. Проверить возможности обучения по правилу Розенблатта.

3. Для второй обучающей выборки построить и обучить сеть, которая будет правиль- но относить точки к четырем классам. Отобразить дискриминантную линию и проверить качество обучения.

1.1 Входные данные

6.	$\begin{bmatrix} -0.5 & 4.9 & -2.1 & -2.1 & 0 & 1.3 \\ -4 & -1.7 & -4.4 & -4.6 & 2.6 & -4.2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

1.2 Структура сети



```
net =

  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: (none)
   divideParam: (none)
```

```
    divideMode: 'sample'
       initFcn: 'initlay'
    performFcn: 'mae'
  performParam: .regularization, .normalization
     plotFcns: {'plotperform', 'plottrainstate'}
    plotParams: {1x2 cell array of 2 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
```

1.3 Алгоритма Розенблатта

1.3.1

Random Weights:
-0.0292 0.6006

Random Biases :
0.9143

```matlab
% Весовые коэффициенты
net.inputWeights{1,1}.initFcn = 'rands';
% Смещения
net.biases{1}.initFcn = 'rands';
% Инициализация
net = init(net);
disp('Random Weights: ');
disp(net.IW{1,1});
disp('Random Biases : ');
disp(net.b{1});
% Отображение точек и прямой
plotpv(P, T);
line1 = plotpc(net.IW{1}, net.b{1});
set(line1, 'Color', 'g');
grid;
hold on;
```

```
function net = Driver_func(net, P, T, iterations, learningRate)
    for j = 1:iterations
        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} + e * p' * learningRate;
                net.b{1} = net.b{1} + e * learningRate;
            end
        end
        disp(['Iteration: ', num2str(j)]);
        disp(['Error: ', num2str(mae(T - net(P)))]);
        disp(['Weights: ', num2str(net.IW{1,1})]);
        disp(['Biases : ', num2str(net.b{1})]);
        disp(' ');
    end
end
```

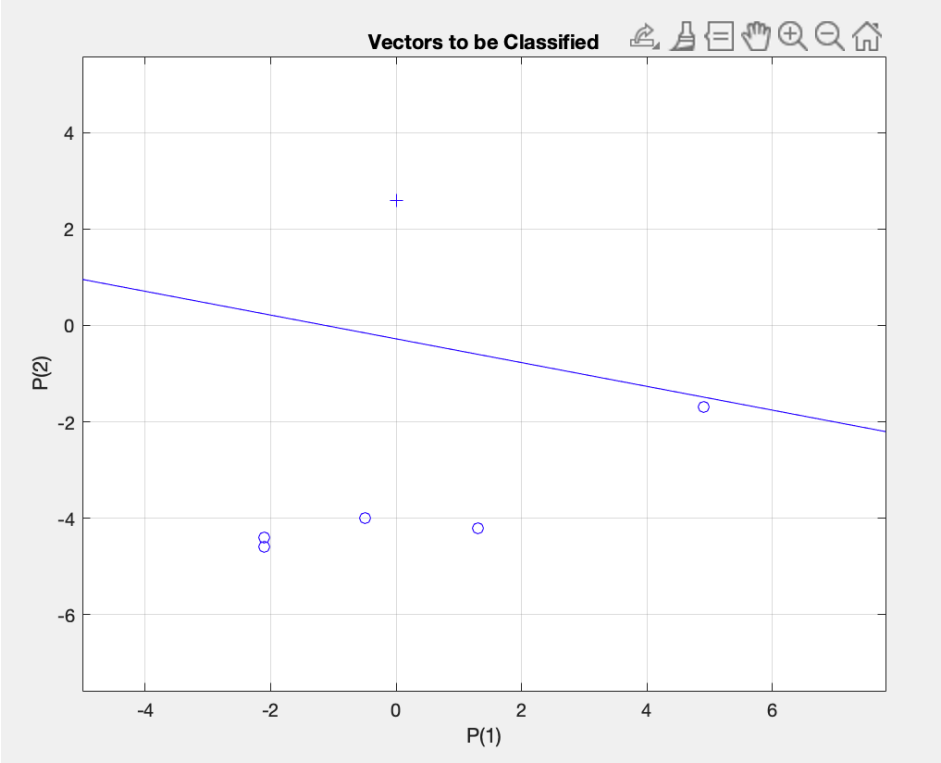
1.3.2 Две итерации

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

Iteration: 1
Error: 0
Weights: 0.32054 0.64762
Biases : -0.61923

Iteration: 2
Error: 0
Weights: 0.32054 0.64762
Biases : -0.61923

1.3.3 Обучающая выборка и дискриминантная линию



1.4.1

Random Weights: 0.73858
0.15941
Random Biases : -0.72786

1.4.2

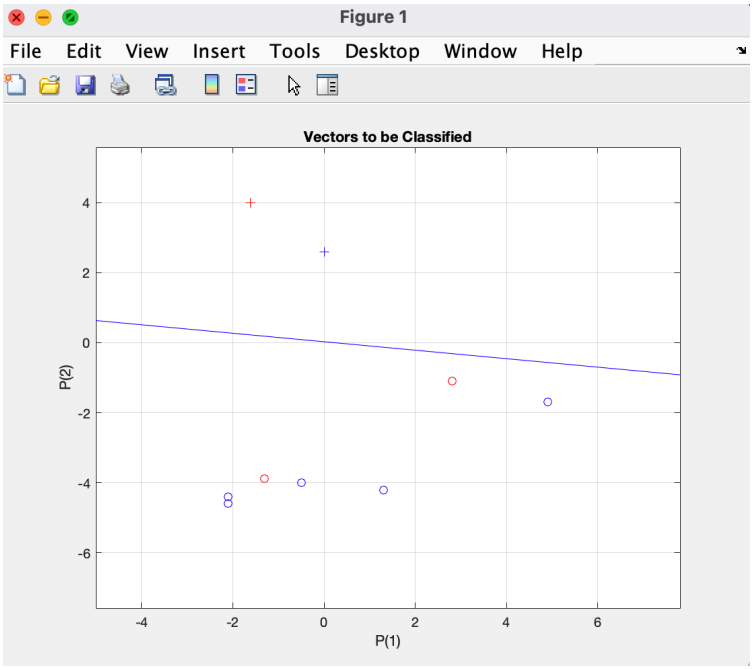
New Weights: -4.1614
1.8594
New Biases : -1.7279

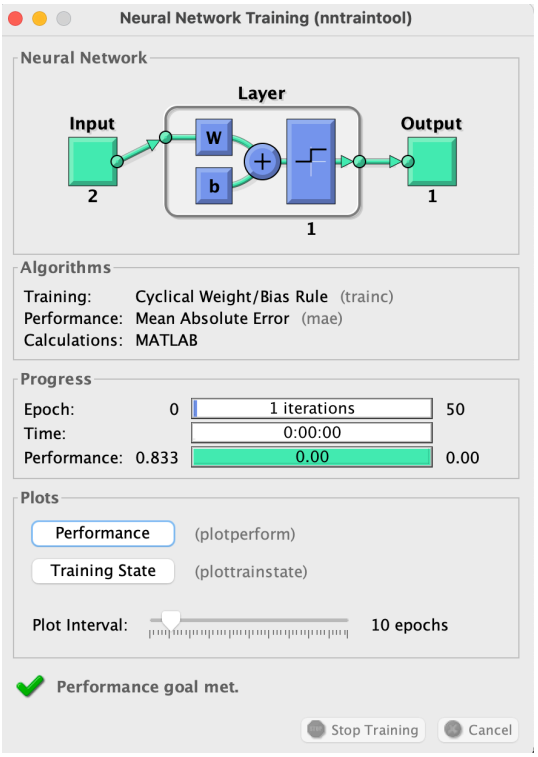
```
% 1.4.1
net = init(net);
disp(['Random Weights: ', num2str(net.IW{1,1})]);
disp(['Random Biases : ', num2str(net.b{1})]);

% 1.4.2
net.trainParam.epochs = 50;
net = train(net, P, T);

disp(['New Weights: ', num2str(net.IW{1,1})]);
disp(['New Biases : ', num2str(net.b{1})]);
```

1.4.3






```
% 1.4.3
% Создаем тестовые точки
testpoints = -5 + (5+5)*rand(2,3);
testclasses = net(testpoints);

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
```

2.1 Линейно неразделимое множество

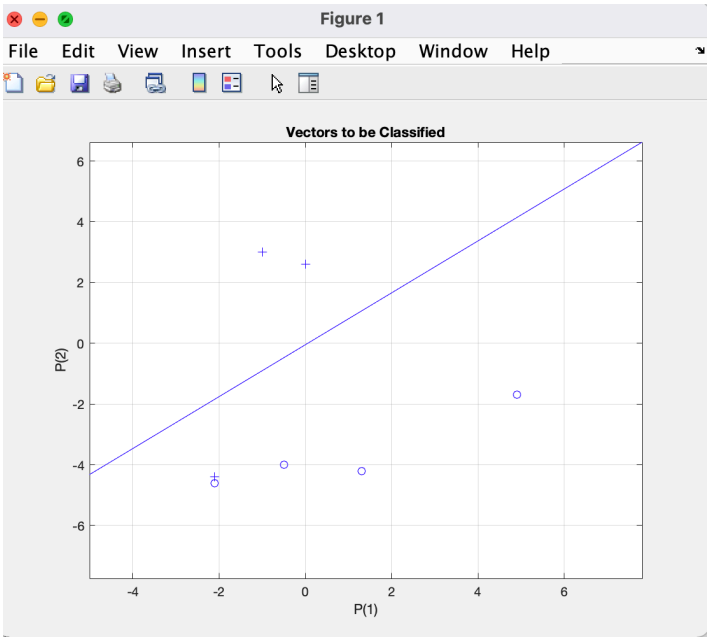
P = [-0.5 4.9 -2.1 -2.1 0 1.3 -1;
-4 -1.7 -4.4 -4.6 2.6 -4.2 3];

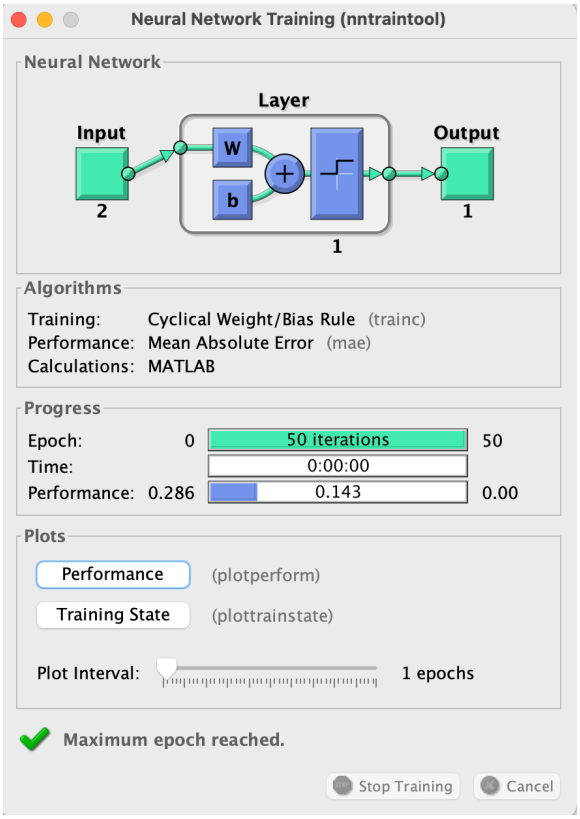
T = [0 0 1 0 1 0 1];

2.2

Random Weights: 0.51548 0.48626
Random Biases : 0.35747

2.3 Результат





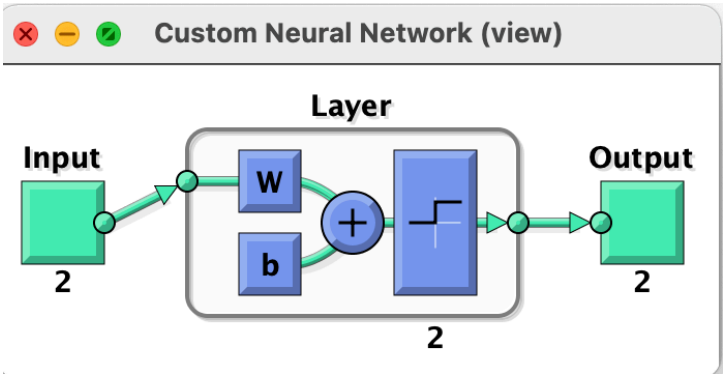
mae:
ans = 0.1429

3.1 Новое множество

P = [-0.8 -2.1 -3.9 2 2.8 -1.1 -2.8 -3.2;
 -2.5 -0.8 -0.1 -2.6 -4.3 -5 -5 -3.6];

T = [1 0 0 1 1 1 0 0;
 0 0 0 0 0 1 1 1];

3.2 Инициализация



3.3 Случайные значения

Random Weights:

-0.5524 -0.4898
0.5025 0.0119

Random Biases:

-0.3192
0.1705

3.4 Обучение

Weights:

5.3476 -2.1898
-2.2975 -3.6881

Biases:

0.6808
-11.8295

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

net.inputWeights{1,1}.initFcn = 'rands';
net.biases{1}.initFcn = 'rands';
net = init(net);

% 3.3
disp('Random Weights: ');
disp(net.IW{1,1});
disp('Random Biases: ');
disp(net.b{1})

% 3.4
net.trainParam.epochs = 50;
net = train(net, P, T);
disp('Weights: ');
disp(net.IW{1,1});
disp('Biases: ')
disp(net.b{1});

plotpv(P, T);
plotpc(net.IW{1}, net.b{1});
grid;

testpoints = -5 + (5+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
```

3.5

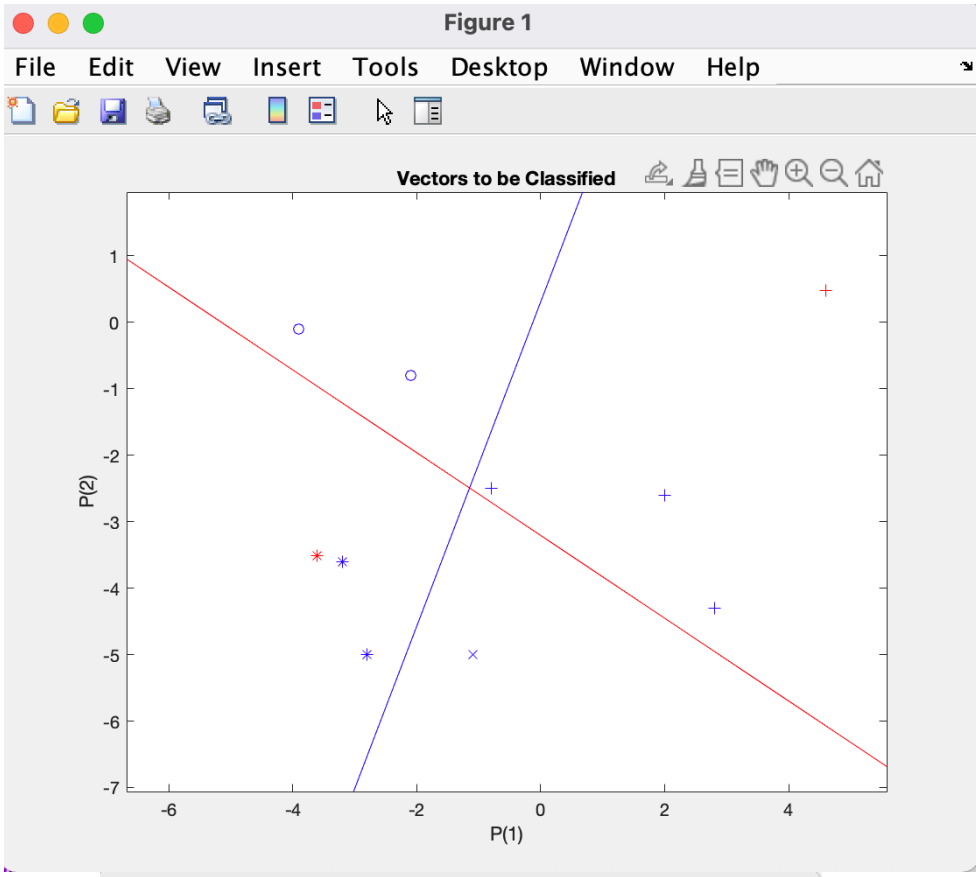
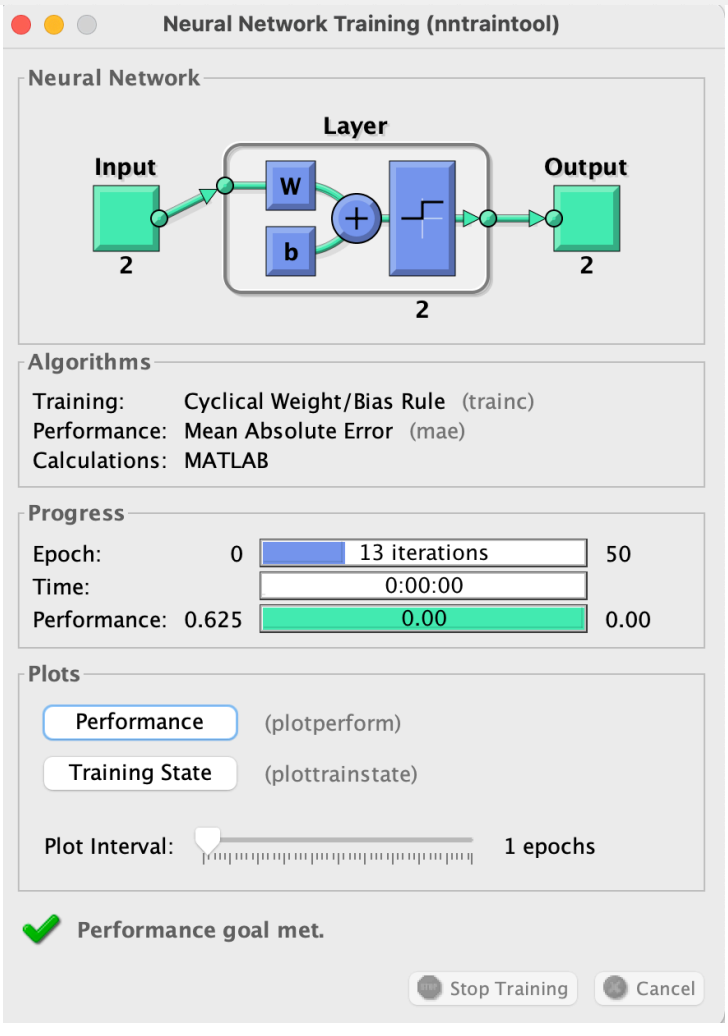


Figure 1
File Edit View Insert Tools Desktop Window Help
Vectors to be Classified
P(2)
P(1)
1
0
-1
-2
-3
-4
-5
-6
-7
-6
-4
-2
0
2
4


Neural Network Training (nntraintool)
Neural Network
Layer
Input
W
b
Output
2
2
2
Algorithms
Training: Cyclical Weight/Bias Rule (trainc)
Performance: Mean Absolute Error (mae)
Calculations: MATLAB
Progress
Epoch: 0 13 iterations 50
Time: 0:00:00
Performance: 0.625 0.00 0.00
Plots
Performance (plotperform)
Training State (plottrainstate)
Plot Interval: 1 epochs
Performance goal met.
Stop Training
Cancel