**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: Куликов А.В.

Группа: М80-408Б-17

Преподаватель: Аносова Н.П.

Дата: 24 сентября 2020

Оценка:

Москва, 2020

**Цель работы:**

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

**Основные этапы работы:**

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

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

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

**Оборудование:**

Процессор: AMD Ryzen 5 Mobile 3550H

Объем оперативной памяти: 8 Гб

**Программное обеспечение:**

MATLAB v9.8.0.1323502 (R2020a)

**Сценарий выполнения работы:**

### Этап №1

P = [-2.8 -0.2 2.8 -2.1 0.3 -1;

1.4 -3.5 -4 -2.7 -4.1 -4];

T = [0 1 1 0 1 0];

Обучающее множество (вариант 8)

net = newp([-5 5; -5 5],[0,1]);

display(net);

net =

Neural Network

name: 'Custom Neural Network'

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

Реализация алгоритма обучения по правилу Розенблатта

function net = rosenblatt\_rule(net, P, T, epochs)

IW\_ = net.IW{1,1};

b\_ = net.b{1,1};

fprintf(repmat('IW[%d]\t\t', 1, size(IW\_,2)), 1:size(IW\_,2));

fprintf(repmat('b[%d]\t\t', 1, size(b\_, 2)), 1:size(b\_,2));

fprintf("MAE\n");

for i = 1:epochs

was\_error = 0;

for j = 1:(size(P, 2))

IW\_ = net.IW{1,1};

b\_ = net.b{1,1};

p = P(:,j);

e = T(:,j) - net(p);

if mae(e)

was\_error = 1;

IW\_ = IW\_ + e \* p.';

b\_ = b\_ + e;

net.IW{1,1} = IW\_;

net.b{1,1} = b\_;

end

end

if was\_error == 0

break;

end

fprintf(repmat('%f\t', size(IW\_)), IW\_);

fprintf(repmat('%f\t', size(b\_)), b\_);

fprintf("%f\n", mae(T - net(P)));

end

end

Инициализация сети случайными значениями

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

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

net = init(net);

Обучение по правилу Розенблатта (2 эпохи)

epochs=2;

net = rosenblatt\_rule(net, P, T, epochs);

IW[1] IW[2] b[1] MAE

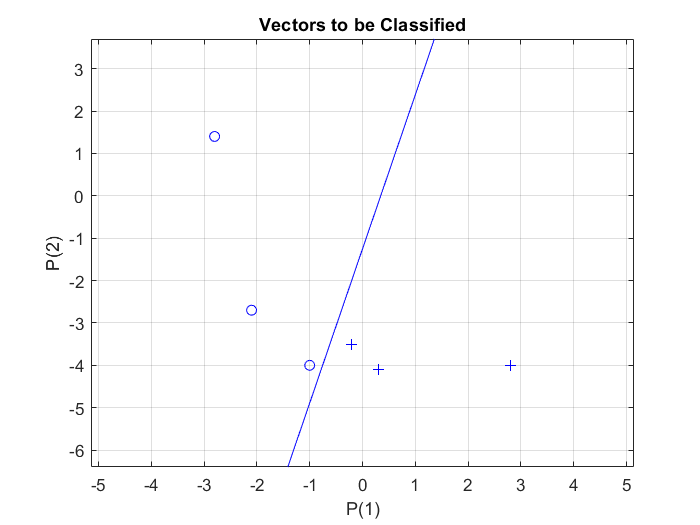
3.412699 2.617017 -2.106433 0.500000

3.212699 -0.882983 -1.106433 0.000000

Отображение дискриминантной линии и обучающей выборки

plotpv(P,T), grid

plotpc(net.IW{1},net.b{1})



Повторная инициализация сети случайными значениями. Обучение сети (50 эпох)

net = init(net);

net.trainParam.epochs = 50;

net = train(net, P, T);

Проверка качества обучения

clf;

plotpv(P,T), grid

plotpc(net.IW{1},net.b{1})

sample = 5 \* rands(2,3);

hold on

red\_color=[1 0 0];

for i=1:size(sample,2)

s = sample(:,i);

if net(sample(:,i))

scatter(s(1), s(2), 10, red\_color, '+');

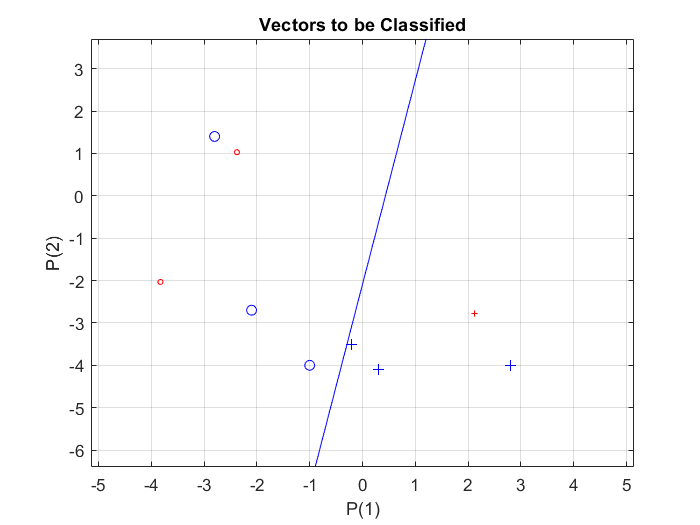
else

scatter(s(1), s(2), 10, red\_color, 'o');

end

end

hold off



### Этап №2

Обучающее множество с линейно неразделимыми классами

P = [-2.8 -0.2 2.8 -2.1 0.3 -1;

1.4 -3.5 -4 -2.7 -4.1 -4];

T = [0 1 1 0 0 1];

Инициализация случайными значениями

net = init(net);

Обучение сети (50 эпох)

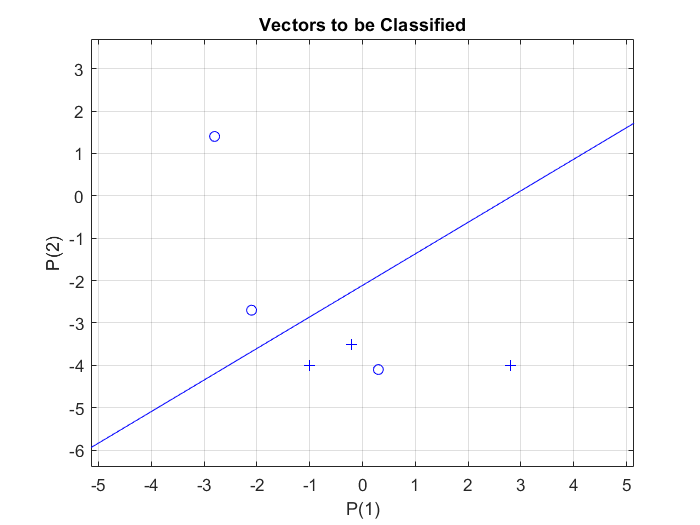
net.trainParam.epochs = 50;

net = train(net, P, T);

Обучающая выборка и дискриминантная линия в случае линейной неразделимости классов

plotpv(P,T), grid

plotpc(net.IW{1},net.b{1})



### Этап №3

Обучающее множество

P = [1.7 4.7 -0.5 1.8 1.5 -1.3 -3.9 4.7;

3.3 -4.5 0.8 2.1 2.2 0.8 -4.5 -2.2];

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

1 1 0 1 1 0 0 1];

Конфигурация сети на 4 класса

net = newp([-5 5; -5 5],[0 1; 0 1]);

display(net);

net =

Neural Network

name: 'Custom Neural Network'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 1

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 6

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

Инициализация случайными значениями. Обучение сети (50 эпох)

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

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

net = init(net);

net.trainParam.epochs = 50;

net = train(net, P, T);

Проверка качества обучения

plotpv(P,T), grid

plotpc(net.IW{1},net.b{1})

sample = 5 \* rands(2,5);

hold on

red\_color=[1 0 0];

for i=1:size(sample,2)

s = sample(:,i);

a = net(s);

if isequal(a, [0; 0])

scatter(s(1), s(2), 10, red\_color, 'o');

elseif isequal(a, [0; 1])

scatter(s(1), s(2), 10, red\_color, '\*');

elseif isequal(a, [1; 0])

scatter(s(1), s(2), 10, red\_color, '+');

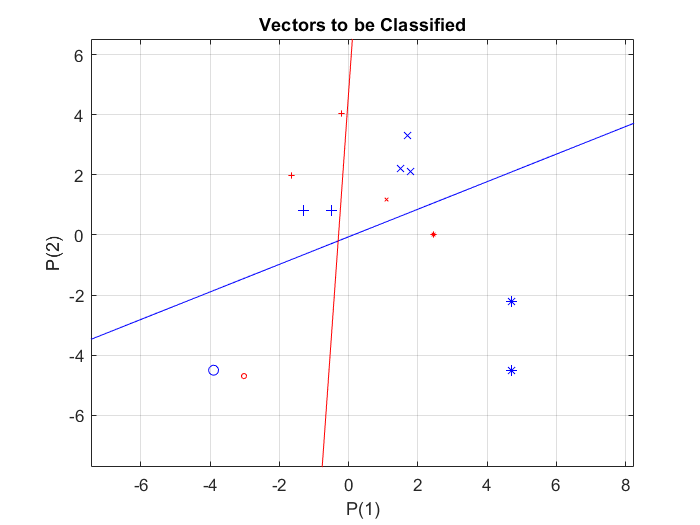
else

scatter(s(1), s(2), 10, red\_color, 'x');

end

end

hold off



**Выводы:**

Перцептрон Розенблатта – первая модель искусственной нейронной сети. Он состоит из одного нейрона, на вход которому подается вектор входных сигналов далее подсчитывается взвешенная сумма входных сигналов c некоторым смещением, к которой применяется передаточная функция (пороговая, в данном случае). На выходе после передаточной функции имеем выходной сигнал, который и есть результат классификации. Перцептрон Розенблатта, несмотря на свою простоту, может решать простые задачи классификации, если классы линейно разделимы. В реальных же задачах, к сожалению, классы не разделимы.

Основные сложности при выполнении работы возникли в освоении пакета MATLAB.