САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет

Лабораторная 1

Исследование НС с пороговой функцией активации (персептронов)

Дисциплина
«Нейроинформатика»

выполнил:
Косенков М.А.
группа: 33531/2
преподаватель:
Никитин К.В.

Санкт-Петербург
2018

1. Оглавление

2. Задание 1.....3
3. Задание 2.....10
4. Задание 3.....15
5. Задание 4.....22
6. Вывод.....27

2. Задание 1

Задание

1. Сформируйте набор входных и желаемых выходных образов (“0” – класс 1, “X” – класс 2. Постройте график и нанесите на него эти точки.

2. Для заданного варианта сформируйте собственный вариант решения задачи без ошибок и реализуйте его аналитически вначале математически с помощью формул, а затем с помощью 2-х или 3-х слойного персептрона (для этого вначале с помощью кусочно-линейной аппроксимации задайте функции, определяющие принадлежность к классу 1 и 2).

3. Сгенерируйте схему командой gensim (net), проанализируйте ее и приведите в отчет.

4. Проверьте полученные функции в заданных точках, а также в близлежащих точках.

5. Попытайтесь решить задачу распознавания путем обучения 1-слойного персептрона на множестве входных примеров. Проанализируйте полученный результат.

Пункт 1

Рассмотрим некоторые точки (*p*), их принадлежность к классам (*p_res*).

```
p = [
.125 .375 .625 .875 .125 .375 .625 .875 .125 .375 .625 .875 .125 .375 .625 .875;
.125 .125 .125 .125 .375 .375 .375 .375 .625 .625 .625 .625 .875 .875 .875 .875;
];

p_res = [1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1];
```

Нанесем на график (Результат представлен на Рис. 1.1.1).

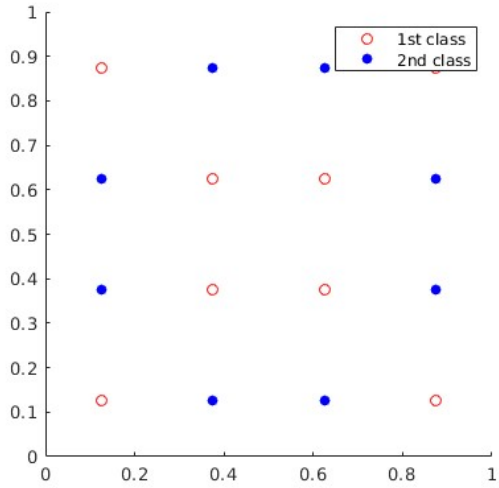

Рис. 1.1.1.

Пункт 2

Реализация в *Matlab* с помощью 3-х слойного персептрона.

Для реализации был написан файл *task1*, в котором строится 3-х слойная нейронная сеть и задаются все необходимые параметры, так же в данном файле происходит попытка обучения однойслойного персептрона. Для построения графиков была написана вспомогательная функция *plot2classes*.

*Файл **task1***

```
net = network(1, 3, [1; 1; 1], [1; 0; 0], [0 0 0; 1 0 0; 0 1 0], [0 0 1]);
net.inputs{1}.size = 2;
net.layers{1}.size = 8;
net.layers{2}.size = 4;
net.layers{3}.size = 1;
net.IW{1, 1} = [
1 1 -1 -1 0 0 0 0;
0 0 0 0 1 1 -1 -1
]';
net.LW{2, 1} = [
0 0 1 0 1 0 0 1;
1 0 0 1 0 1 0 0;
0 1 0 0 1 0 0 1;
1 0 0 1 0 0 1 0;
];
net.LW{3, 2} = [1 1 1 1];
net.b{1} = [ -.25 -.75 .25 .75 -.25 -.75 .25 .75]';
net.b{2} = [-2.5 -2.5 -2.5 -2.5]';
net.b{3} = -.5;
net.layers{1}.transferfcn = 'hardlim';
net.layers{2}.transferfcn = 'hardlim';
net.layers{3}.transferfcn = 'hardlim';
gensim(net);
p = [
.125 .375 .625 .875 .125 .375 .625 .875 .125 .375 .625 .875 .125 .375 .625 .875;
.125 .125 .125 .125 .375 .375 .375 .375 .625 .625 .625 .625 .875 .875 .875 .875;
];
p_res = [0 1 1 0 1 0 0 1 1 0 0 1 0 1 1 0];
plot2classes(p', p_res);

X = rand(1000, 2);
test = sim(net, X');
plot2classes(X, test);

 per = newp([0 1; 0 1], 1);
 [per, f] = train(per, X', test);
 per(p);
 r = rand(2000, 2);
 result = per(r');
plot2classes(r, result);
```

*Файл **plot2classes***

```
function plot2classes(X, Y)
num_of_ones = sum(Y(:) == 1);
num_of_zeros = sum(Y(:) == 0);
ones_ = zeros(num_of_ones, 2);
zeros_ = zeros(num_of_zeros, 2);
j = 1;
k = 1;
for i = 1:length(Y)
   if Y(i) == 1
      ones_(j, 1) = X(i, 1);
      ones_(j, 2) = X(i, 2);
      j = j + 1;
   else
      zeros_(k, 1) = X(i, 1);
      zeros_(k, 2) = X(i, 2);
      k = k + 1;
   end
end
figure
hold on
scatter(ones_(:,1), ones_(:,2), 'r');
scatter(zeros_(:,1), zeros_(:,2), 'b', 'filled');
axis([0 1 0 1])
axis square
legend(['1st class'; '2nd class']);
hold off
end
```

Пункт 3

Посмотрим на результат выполнения команды gensim. Команда сгенерировала графическое представление построенной нейронной сети. Результат представлен на Рис 1.3.1 – 1.3.6.

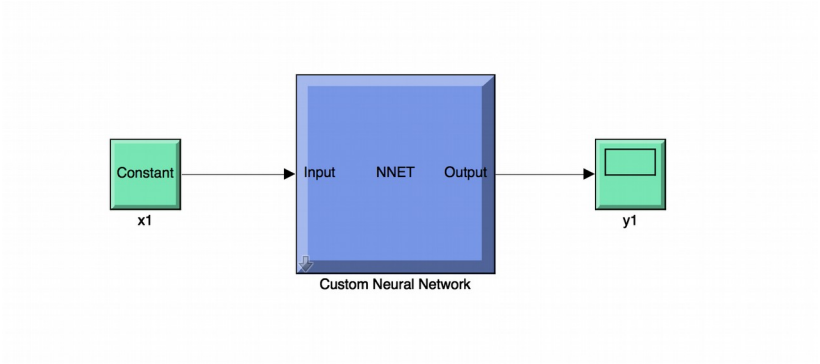

Рис 1.3.1. Общий вид НС.

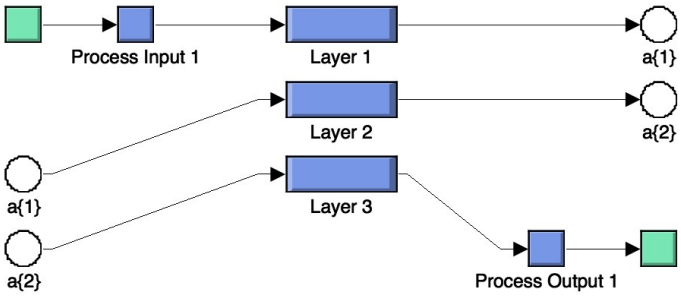

Рис 1.3.2. Общий вид НС на уровне слоев.

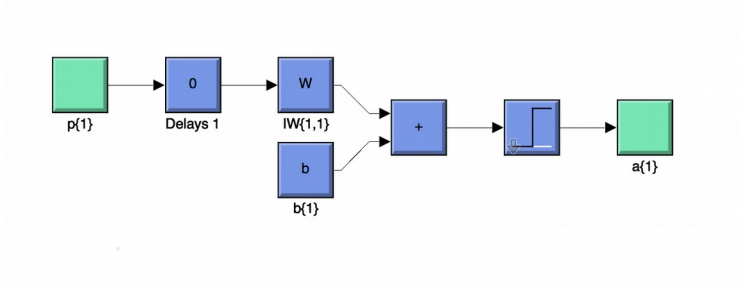

Рис 1.3.3. Общий вид 1-го слоя.

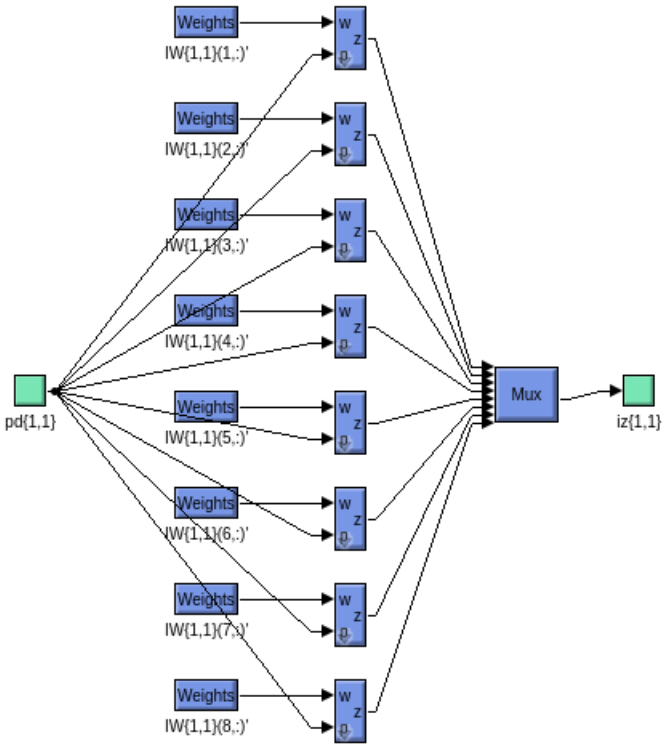

Рис 1.3.4. Применение весов в 1-ом слое.

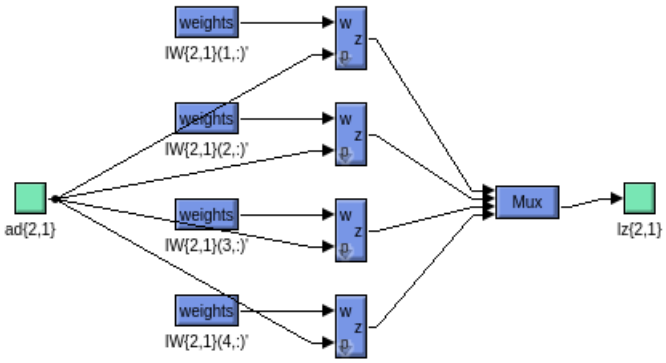

Рис 1.3.5. Применение весов во 2-ом слое.

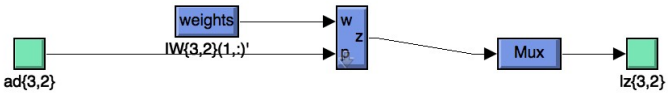

Рис 1.3.6. Применение весов в 3-ем слое.

Пункт 4

Проверка работы нейронной сети на случайном наборе точек. Визуализация результата представлена на Рис 1.4.1.

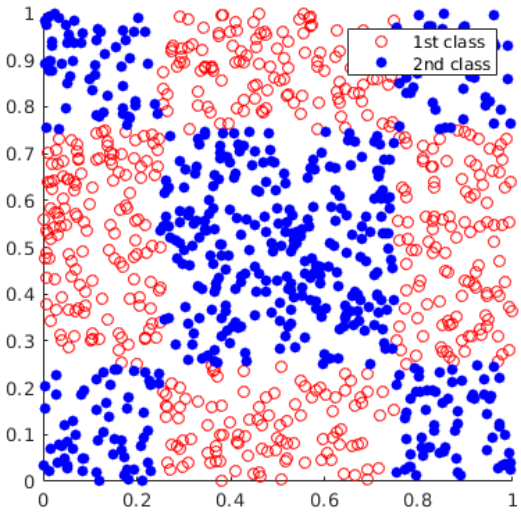

Рис 1.4.1. Результат работы НС

Видим, что НС работает верно, она правильно определяет принадлежность точек к классам.

Пункт 5

Попробуем обучить однослойный персептрон с 1 нейроном на заданном наборе точек. Очевидно, НС не обучится верно, так как однослойный персептрон с 1 нейроном способен реализовать только линейной разделение классов. Окно процесса обучения представлено на Рис 1.5.1, график изменения ошибки представлен на Рис 1.5.2, результат работы персептрона на случайном наборе точек представлен на Рис 1.5.3.

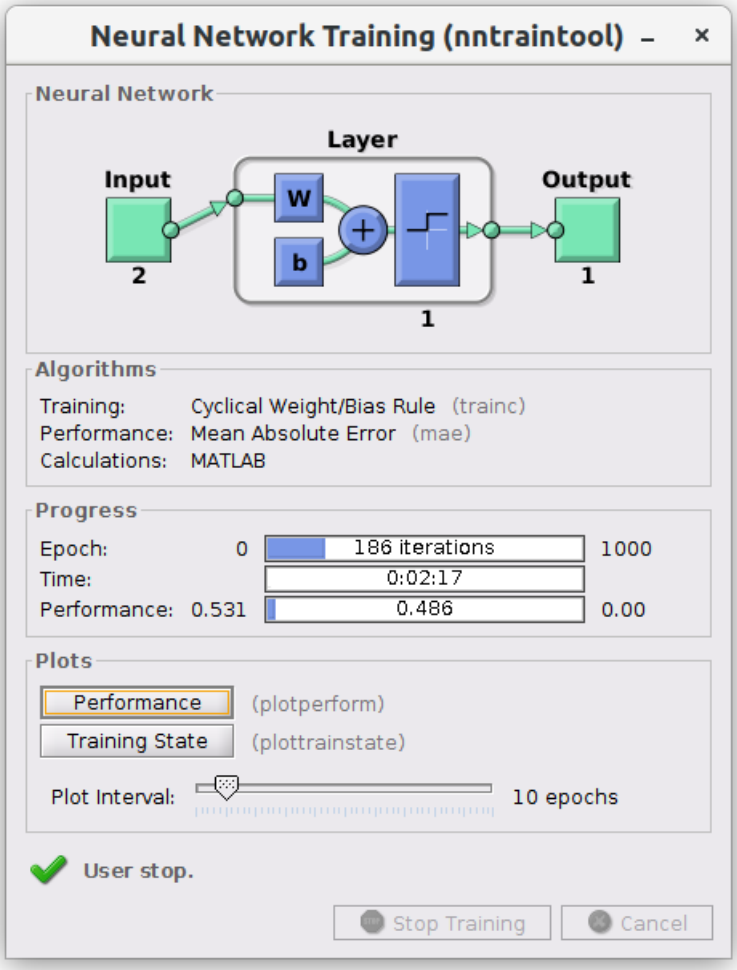

Рис 1.5.1. Процесс обучение персептрона.

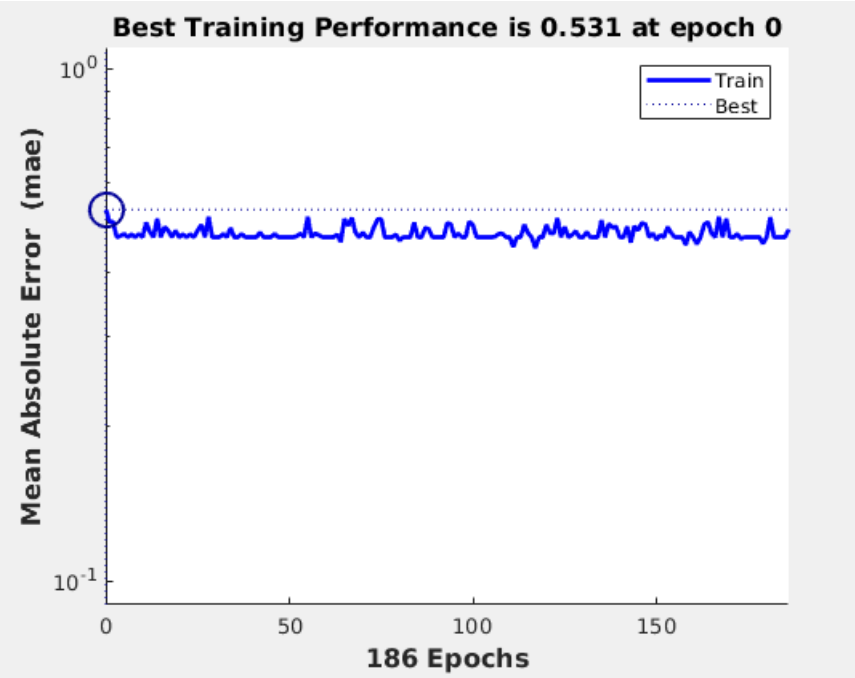

Рис 1.5.2. График изменения ошибки.

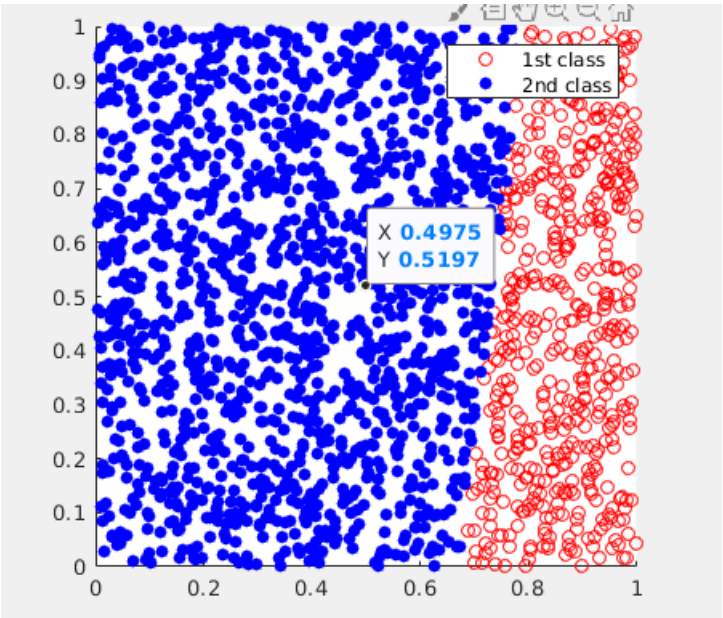

Рис 1.5.3. Рузультат работы персептрона

Процесс обучения был остановлен вручную, так как значение *performance* перестало меняться (точнее менялось, но очень медленно и в разные стороны). По результатам работы персептрона, можно увидеть, что он реализован функцию прямой, разделяющей плоскость на две части. Полученное разделение зависит от обучающего набора точек и времени обучения.

3. Задание 2

Задание

1. Запишите выражение для вашей ЛФ в форме СДНФ.
2. Реализуйте полученную СДНФ в форме 2-слойного персептрона (net) в Matlab.
3. Сгенерируйте схему командой gensim (net), проанализируйте ее и приведите в отчет.
4. Проверьте правильность работы полученной НС по таблице истинности.
5. Попробуйте обучить однослойный персептрон на Вашей функции, используя в качестве обучающей выборки фрагменты таблицы истинности. Проанализируйте результаты и посчитайте среднюю ошибку.

Пункт 1

СДНФ для функции из lab0:

$$\overline{x}_1\overline{x}_2\overline{x}_3x_4\overline{x}_5+\overline{x}_1x_2\overline{x}_3\overline{x}_4x_5+x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5$$
$$+x_1\overline{x}_2x_3\overline{x}_4x_5+x_1x_2\overline{x}_3\overline{x}_4x_5$$

Пункт 2

Реализуем полученную СДНФ в *Matlab*. Для этого нам понадобится 2-ух слойная НС. Первый слой будет реализовывать конъюнкции 5 переменных, а второй слой – диъюнкция полученных от конъюнкций результатов.

Напишем функции для 1-го слоя. Плюс перед переменной означает, что она входит без инверсии, а минус значит, что с инверсией. Прибавляемое число равно ($0.5 - \textit{сумма всех положительных переменных}$.

$$\begin{cases} -x_1-x_2-x_3+x_4-x_5-0.5=0 \\ -x_1+x_2-x_3-x_4+x_5-1.5=0 \\ x_1-x_2-x_3-x_4+x_5-1.5=0 \\ x_1-x_2+x_3-x_4+x_5-2.5=0 \\ x_1+x_2-x_3-x_4+x_5-2.5=0 \end{cases}$$

Пусть выходом 1-ого слоя будет вектор *Z*, каждый элемент которого равен 0 или 1, в зависимости от результата конъюнкций.

Напишем функцию для 2-го слоя. Она будет простая, так как нам необходимо лишь реализовать функцию “ИЛИ” (дизъюнкцию).

$$\sum_{i=1}^{6} z_i - 0.5 = 0$$

Теперь реализуем это все в виде НС в *Matlab*. Для реализации был написан файл *task2*, в котором описана 2-ух слойная НС. Файл с комментариями представлен в директории *programs*.

*Файл **task2***

```
net = network(1, 2, [1; 1], [1; 0], [0 0; 1 0], [0 1]);
net.inputs{1}.size = 5;
net.layers{1}.size = 5;
net.layers{2}.size = 1;
net.IW{1, 1} = [
   -1 -1 -1 1 -1;
   -1 1 -1 -1 1;
   1 -1 -1 -1 1;
   1 -1 1 -1 1;
   1 1 -1 -1 1;
   ];
net.LW{2, 1} = [ 1 1 1 1 1 ];
net.b{1} = [-.5 -1.5 -1.5 -2.5 -2.5]';
net.b{2} = -.5;
net.layers{1}.transferfcn = 'hardlim';
net.layers{2}.transferfcn = 'hardlim';
gensim(net);
p = [
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1;
   0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1;
   0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1;
   0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1;
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1;
];

p_res = sim(net, p);

per = newp([0 1; 0 1; 0 1; 0 1; 0 1], 1);
[per, f] = train(per, p(:,1:20), p_res(1:20));
newp_res = per(p);
errors = 0;
for i=1:length(p_res)
   if (p_res(i) ~= newp_res(i))
      errors = errors + 1;
   end
end

rel_err = errors / length(p_res)
rel_right = (length(p_res) - errors) / length(p_res)
```

Пункт 3

Командой *gensim* была получена структура НС. Результат представлен на Рис 2.3.1 – Рис 2.3.4.

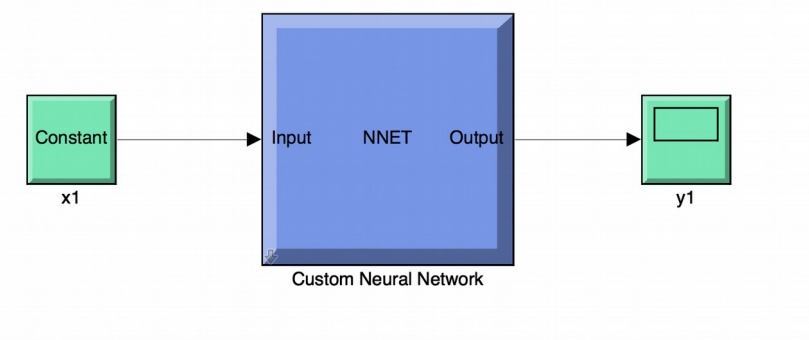

Рис 2.3.1. Общая структура НС.

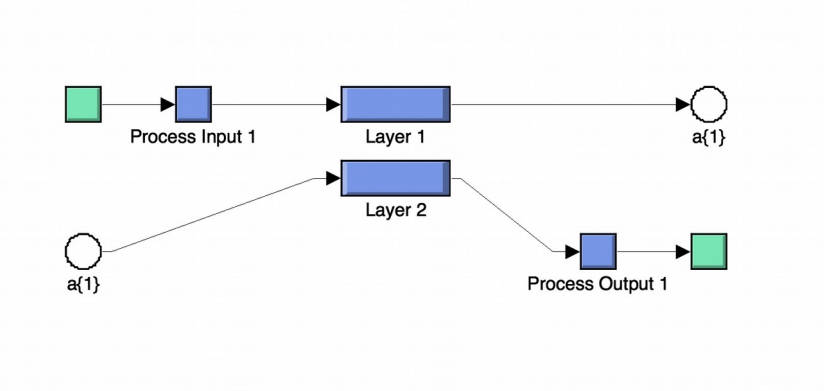

Рис 2.3.2. Структура НС на уровне слоев.

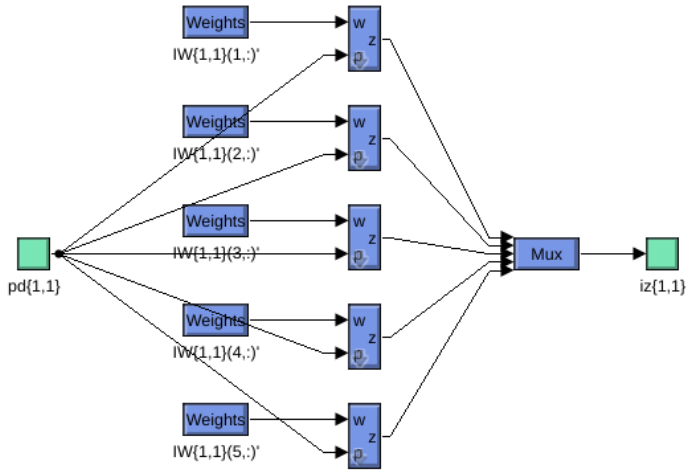

Рис 2.3.3. Применение весов в 1-ом слое.

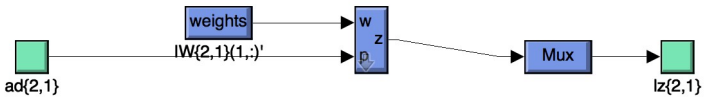

Рис 2.3.4. Применение весов во 2-ом слое.

Пункт 4

Проверим работоспособности НС. Для этого была написана матрица *p* со всеми возможными комбинациями входных параметров, затем результат работы НС был записан в матрицу *p_res*.

Посмотрим на результат *p_res* и сравним его с правильными результатами, представленными в *lab0*.

```
p_res = [     0     0     1     0     0     0     0     0     0     1     0
0     0     0     0     0     0     1     0     0     0     1     0     0
0     1     0     0     0     0     0     0     ]
```

Сравнив с правильными результатами, выясняем, что НС работает безошибочно.

Пункт 5

Попробуем обучить однойслойный персептрон. Для этого был сгенерирован персептрон с одним нейроном и 5 входами.

Окно процесса обучения представлено на Рис 2.5.1, а график изменения ошибки на Рис 2.5.2

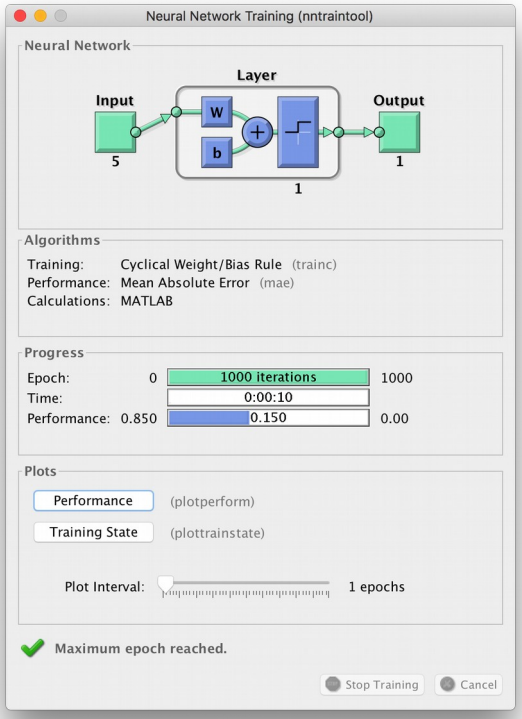

Рис 2.5.1. Окно процесса обучения персептрона.

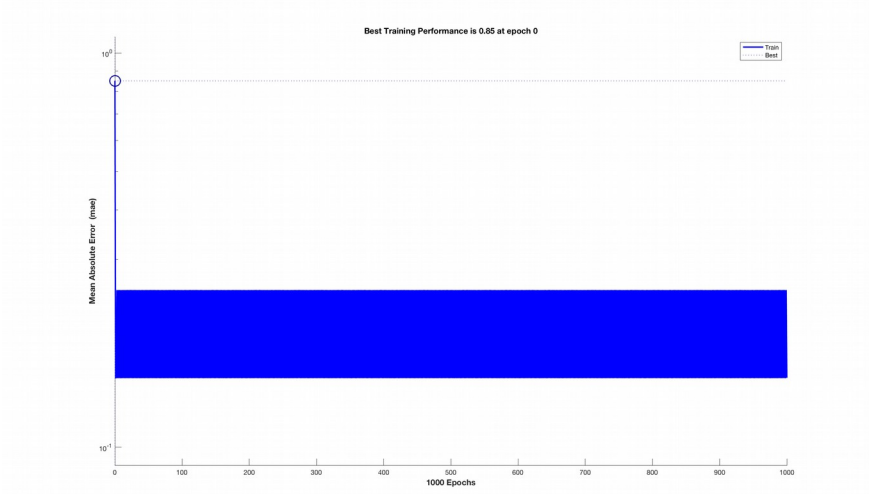

Рис 2.5.2. График изменения ошибки.

Персептрон корректно обучился, так как одного слоя для наших целей явно недостаточно. Можно было бы построить персептрон с 32 нейронами, но тогда он бы просто запоминал правильные ответы.

```
newp_res = [ 0     1     0     0     0     0     0     0     0     0     0
0     0     0     0     0     1     1     0     0     0     1     0     0
0     1     0     0     0     0     0     0     ]
```

В итоге процент верных ответов получился довольно высоким $rel_{rig}ht=0.875$, а ошибок – $rel_{err}=0.125$. Но такой результат связан лишь с тем, что большая часть

ответов, должна была быть 0. Если учитывать только те комбинации, когда на выходе должна появляться единицы, то процент ошибок будет $100\,\%$.

4. Задание 3

Задание

1. Запишите аналитическое выражение для функции, реализующей Ваше разбиение плоскости на 2 класса. Постройте график функции и разбиение плоскости на классы.
2. Реализуйте полученную функцию в форме 2-х или 3-х слойного персептрона net в Matlab.
3. Сгенерируйте схему командой gensim (net), проанализируйте ее и приведите в отчет.
4. Проверьте правильность работы полученной НС путем построения разбиения плоскости на классы, которое реализует персептрон.
5. Попытайтесь решить задачу распознавания путем обучения 1-слойного персептрона на множестве входных примеров. Для этого вначале сформируйте обучающую выборку необходимого объема.
6. После обучения посчитайте среднюю ошибку, проанализируйте результаты – какую функцию реализует обученная сеть.

Пункт 1

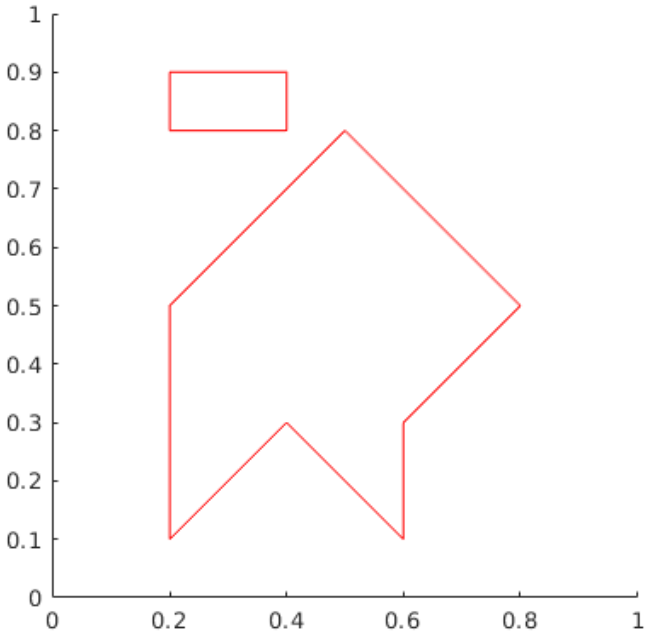

Рис 3.1.1. Разбиение плоскости на 2 класса.

Пункт 2

Решение задачи классификации данной плоскости было реализовано в Matlab в виде трехслойного персептрона с 14нейронами в 1-ом слое, 4 нейронами во 2-ом слое и 1 нейроном в 3-ем слое. На вход НС принимает двумерный вектор.

*Файл **task3***

```
net = network(1, 3, [1; 1; 1], [1; 0; 0], [0 0 0; 1 0 0; 0 1 0], [0 0 1]);
net.inputs{1}.size = 2;
net.layers{1}.size = 14;
net.layers{2}.size = 4;
net.layers{3}.size = 1;
net.IW{1, 1} = [
1 -1 0 0 1 -1 -1 -1 1 1 -1 -1 1 1 ;
0 0 1 -1 0 1 -1 1 -1 1 -1 0 -1 1
]';

net.LW{2, 1} = [
1.0 0.0 0.0 0.0 ;
1.0 0.0 0.0 0.0 ;
1.0 0.0 0.0 0.0 ;
1.0 0.0 0.0 0.0 ;
0.0 1.0 0.0 0.0 ;
0.0 1.0 0.0 0.0 ;
0.0 1.0 0.0 0.0 ;
0.0 0.0 1.0 0.0 ;
0.0 0.0 1.0 0.0 ;
0.0 0.0 1.0 0.0 ;
0.0 0.0 1.0 0.0 ;
0.0 0.0 0.0 1.0 ;
0.0 0.0 0.0 1.0 ;
0.0 0.0 0.0 1.0 ;
]';
net.LW{3, 2} = [1 1 1 1];
net.b{1} = [-0.2 0.4 -0.8 0.9 -0.2 0.1 0.7 0.3 0.3 -0.7 1.3 0.6 -0.3 -0.7]';
net.b{2} = [-3.5 -2.5 -3.5 -2.5]';
net.b{3} = -.5;
net.layers{1}.transferfcn = 'hardlim';
net.layers{2}.transferfcn = 'hardlim';
net.layers{3}.transferfcn = 'hardlim';
gensim(net);
X = rand(2000, 2);
test = sim(net, X');

plot2classes(X, test);
subX = X(1:1000,:);

subT = test(1:1000);
subX2 = X(1000:end,:);
subT2 = test(1000:end);
```

```
per = newp([0 1; 0 1], 1);
per= init(per);
[per, f] = train(per, subX', subT);
result = per(subX2');
plot2classes(subX2, result);

error = 0;
for i = 1:1000
if (subT2(i) ~= result(i))
error = error + 1;
end
end

error / length(subT2)
```

Пункт 3

Схема, сгенерированная командой *gensim* представлена на Рис 3.3.1 – 3.3.5.

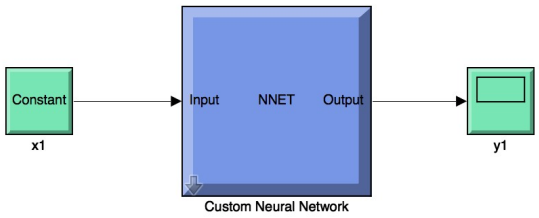

Рис 3.3.1. Общий вид НС.

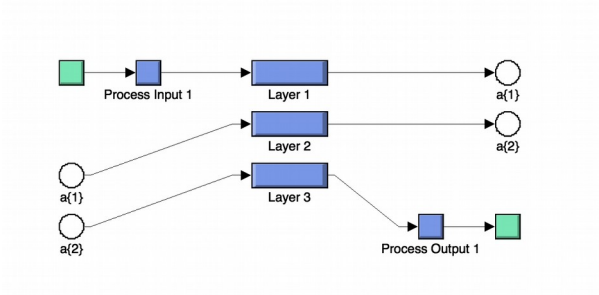

Рис 3.3.2. Общий вид НС на уровне слоев.

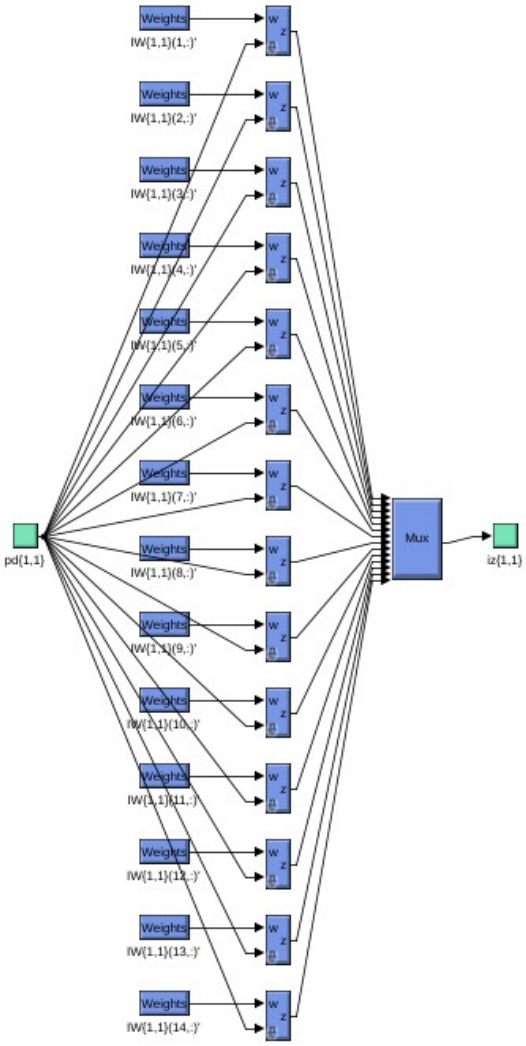

Рис 3.3.3. Весовые коэффициенты для 1-го слоя.

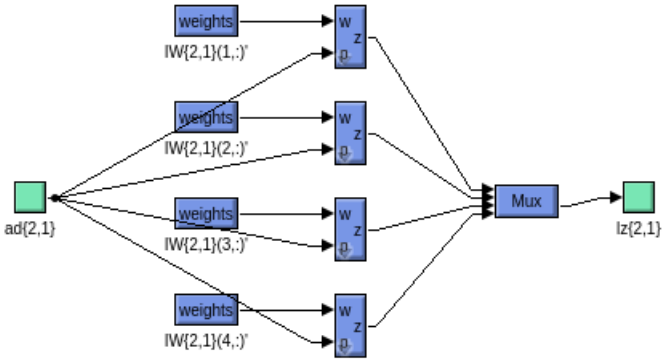

Рис 3.3.4. Весовые коэффициенты для 2-го слоя.

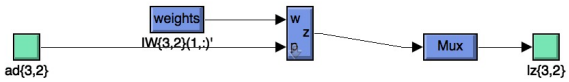

Рис 3.3.5. Весовые коэффициенты для 3-го слоя.

Пункт 4

После построение НС, была сгенерирована случайная выборка из 2000 точек, на которой была протестирована НС. Результаты теста можно увидеть на Рис 3.4.1.

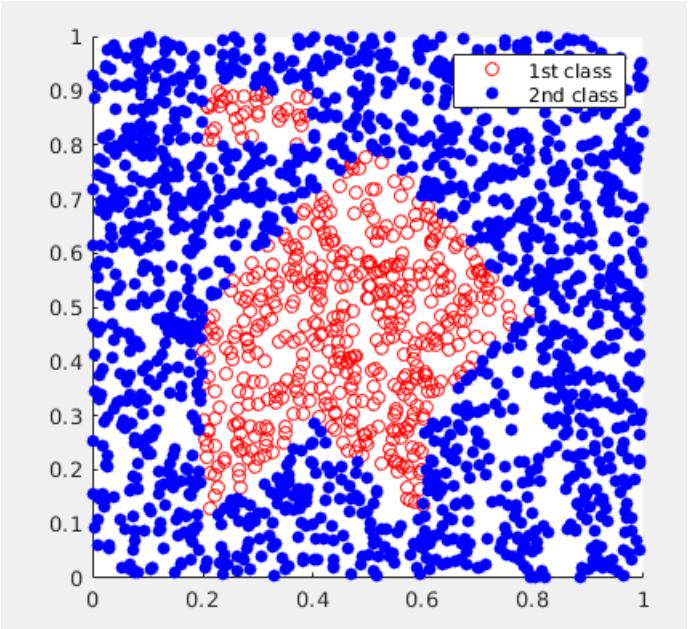

Рис 3.4.1. Результат классификации НС.

Как видим, НС успешно справилась с поставленной задачей.

Пункт 5, 6

Для обучения однослойного персептрона была взята половина той выборки, которая использовалась для тестирования собранной вручную НС, вторая же половини использовалась для тестирования персептрона. Процесс обучения представлен на Рис 3.5.1, график изменения ошибки на Рис 3.5.2, результат классификации на Рис 3.5.3.

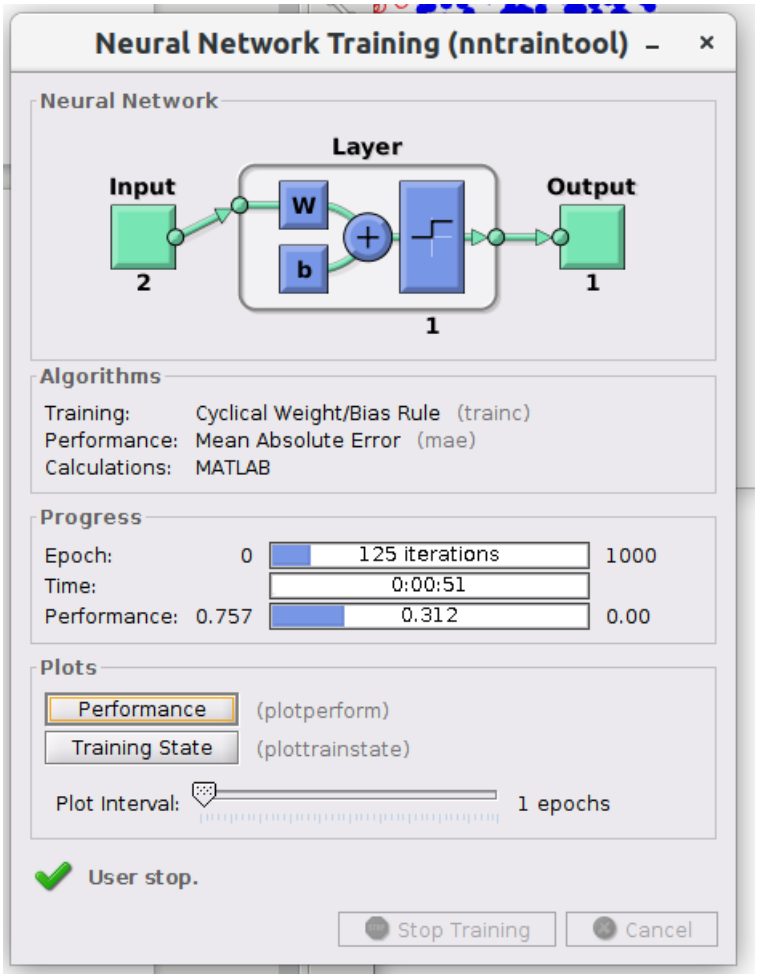

Рис 3.5.1. Процесс обучения персептрона.

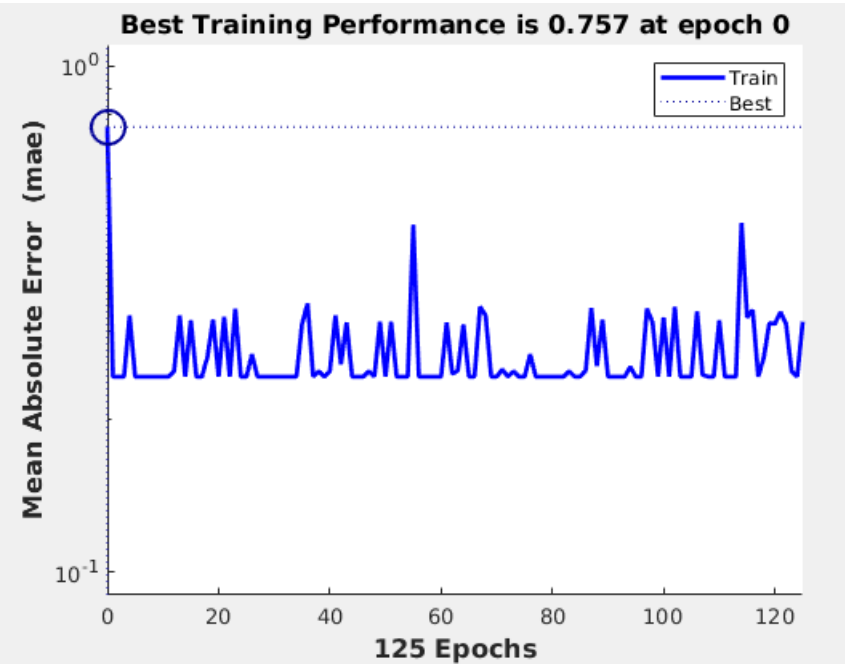

Рис 3.5.2. График изменения ошибки.

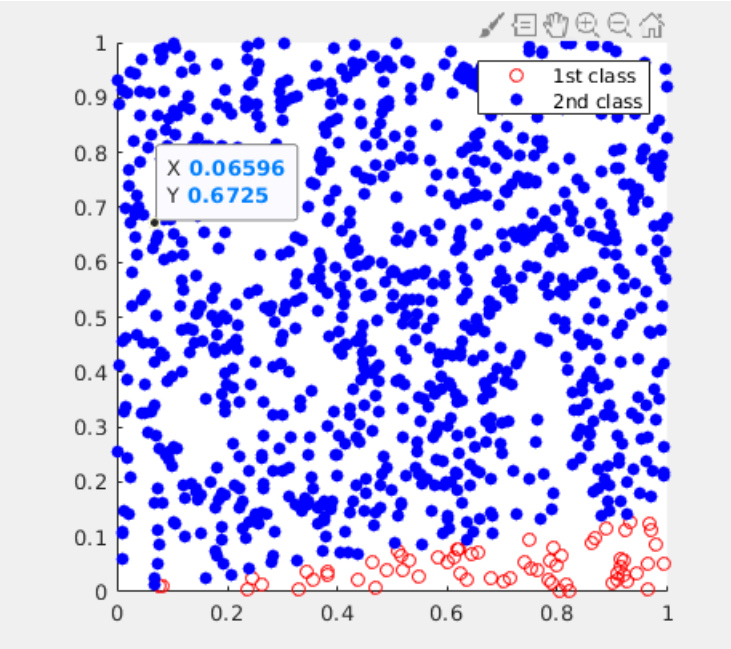

Рис 3.5.3. Результат классификации персептроном.

Как видим, персептрон не смог решить задачу. Он выполняет деление плоскости на линейно разделимые классы. Ошибка при этом получилась 29.17%. Однако это цифра зависит от данных и времени обучения, так как сама линия разделения постоянно меняет свое положение.

5. Задание 4

Задание

1. Запишите аналитическое выражение для функции, реализующей Ваше разбиение плоскости на m классов. Постройте график с разбиением плоскости на m классов.
2. Реализуйте полученную функцию в форме 2-х или 3-х слойного персептрона net в Matlab.
3. Сгенерируйте схему командой gensim (net), проанализируйте ее и приведите в отчет.
4. Проверьте правильность работы полученной НС путем построения разбиения плоскости на классы, которое реализует персептрон.

Пункт 1

Возьмем разбиение плоскости на 5 классов из *lab0*.

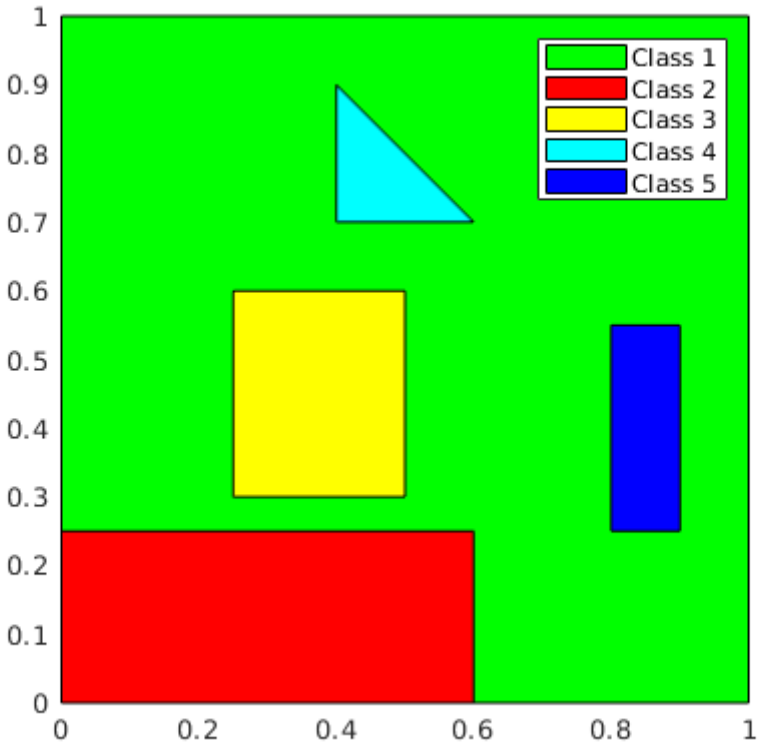

Рис 4.1.1. Разбиение плоскости на 5 классов.

Пункт 2

Написали программу *task4*. В ней использовали программы *is_in_area2* и *plot5classes* из *lab0*. В программе происходит построение НС, сравнение результатов полученных НС и функцие *in_in_area2*. В конце программа рисует график.

Программа ***task4***

```
net = network(1, 3, [1; 1; 1], [1; 0; 0], [0 0 0; 1 0 0; 0 1 0], [0 0 1]);
net.inputs{1}.size = 2;
net.layers{1}.size = 13;
net.layers{2}.size = 4;
net.layers{3}.size = 5;
net.IW{1, 1} = [
   1 -1 1 -1 1 -1 0 0 0 0 0 0 -1;
   0 0 0 0 0 0 1 -1 1 -1 1 -1 -1
   ]';
net.LW{2, 1} = [
   1 1 0 0 0 0 1 1 0 0 0 0 0;
   0 0 1 1 0 0 0 0 1 1 0 0 0;
   0 0 0 0 1 0 0 0 0 0 1 0 1;
   0 0 0 0 0 1 0 0 0 0 0 1 0;
];
net.LW{3, 2} = [
   1 0 0 0 0;
   0 1 0 0 0;
   0 0 1 0 0;
   0 0 0 1 0;
]';
net.b{1} = [-.3 .5 -.8 .9 -.4 .6 -.3 .6 -.25 .55 -.7 .25 1.3 ]';
net.b{2} = [-3.5 -3.5 -2.5 -1.5]';
net.b{3} = [-.5 -.5 -.5 -.5 -.5]';
net.layers{1}.transferfcn = 'hardlim';
net.layers{2}.transferfcn = 'hardlim';
net.layers{3}.transferfcn = 'hardlim';
gensim(net);
r = rand(10000, 2);
test = sim(net, r');
real = is_in_area2(r, 2);
error = 0;
for i = 1:length(r)
   if (~isequal(test(:,i), real(:,i))) error = error + 1;
   end
end
plot5classes(r, test);
```

Пункт 3

Схема, полученная командой *gensim,* представлена на Рис 4.3.1 – 4.3.5.

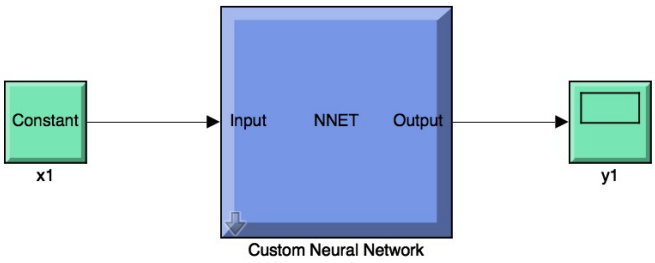

Рис 4.3.1. Общий вид НС.

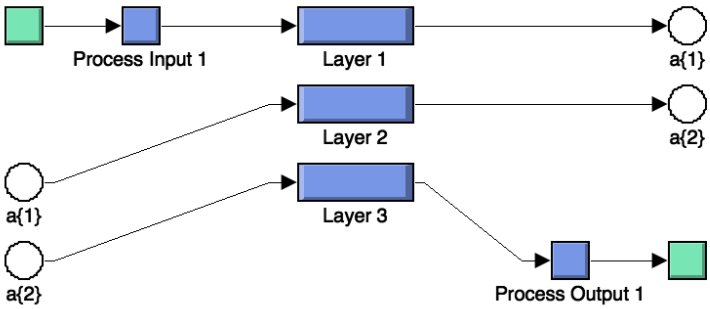

Рис 4.3.2. Общий вид НС на уровне слоев.

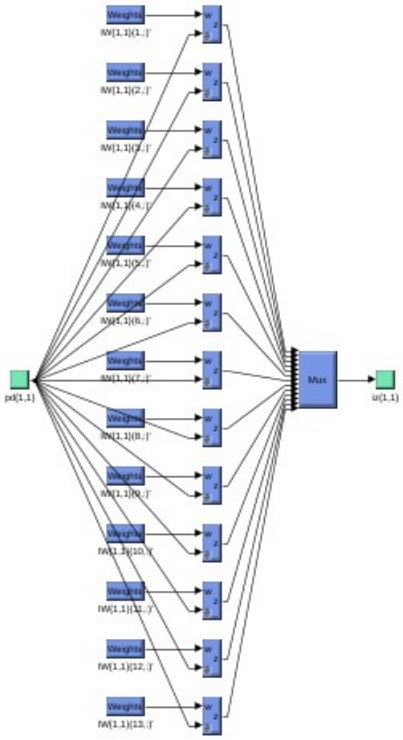

Рис 4.3.3. Веса для 1-го слоя.

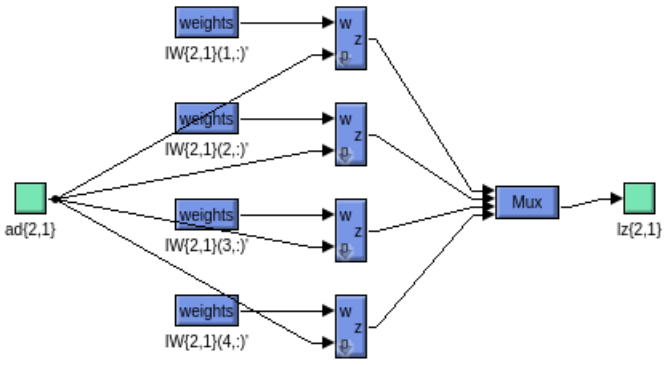

Рис 4.3.4. Веса для 2-го слоя.

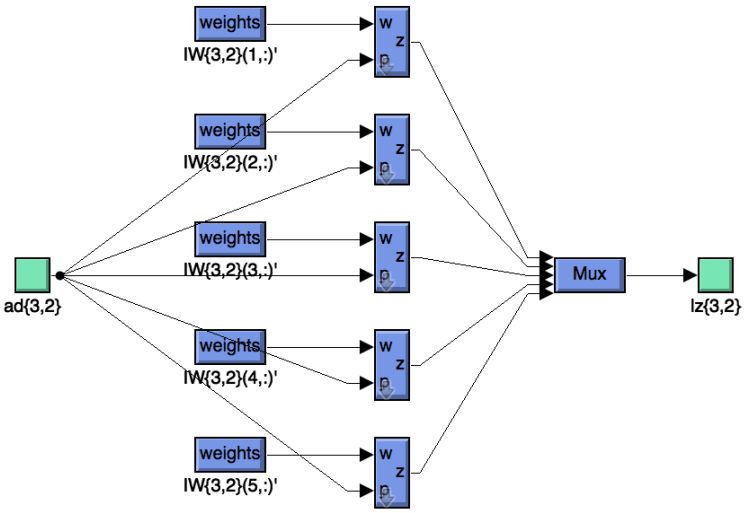

Рис 4.3.5. Веса для 3-го слоя.

Пункт 4

Ошибка равна 0. График, показывающий как НС распознала точки представлен на Рис 4.4.1.

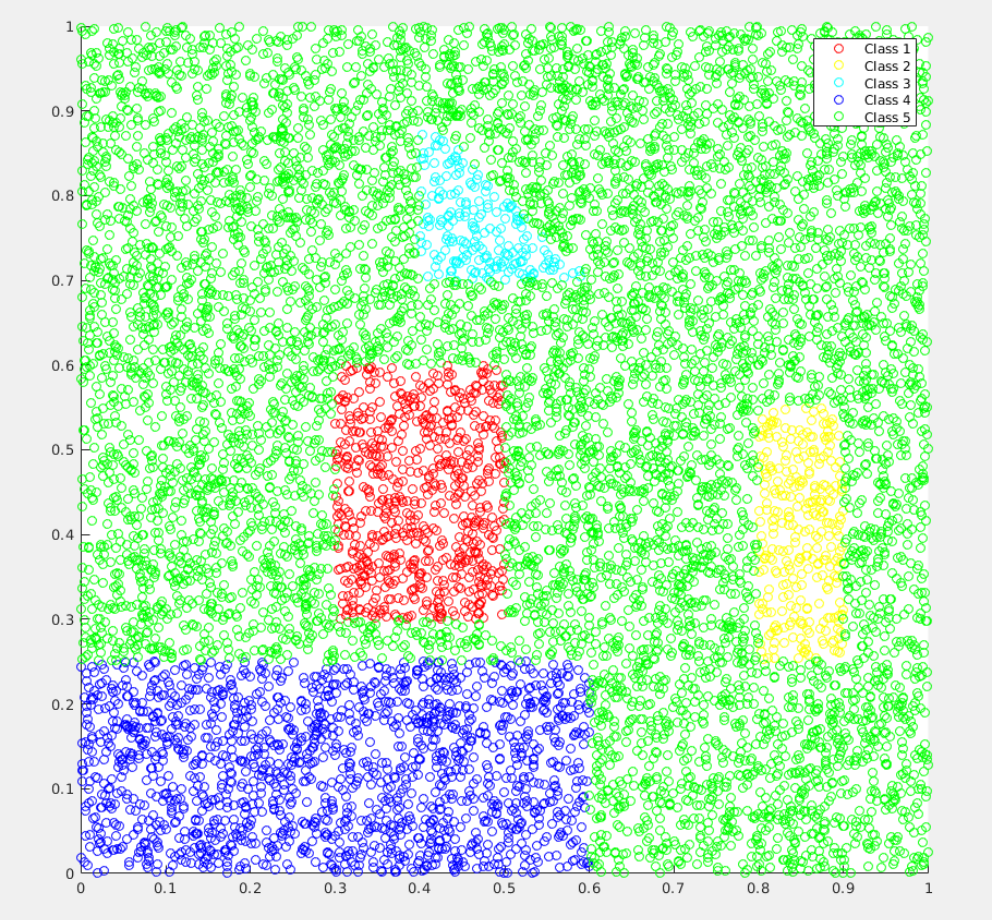

Рис 4.4.1. Результат классификации.

Задание 5

Задание

1. Сгенерировать две выборки X-Y1, X-Y2, представляющих собой множество двух линейно разделимых и линейно-неразделимых классов на плоскости.

а. Сгенерировать множество равномерно случайно распределенных точек X в квадрате с углами (0,0) и (1,1).

б. Провести через квадрат прямую линию y = k x + b так, чтобы с обеих сторон линии было примерно одинаковое количество точек.

Зам. Значения k и b должны быть уникальными.

Промаркировать точки с одной стороны линии как относящиеся к первому классу, а с другой стороны – ко второму классу. Запомнить разбиение точек как X-Y1.

в. Провести через квадрат кривую линию, разделяющую его на две части такую, чтобы:

- в каждой части было примерно одинаковое число точек
- невозможно было провести прямую линию, разделяющую квадрат на такие же два множества точек.

Например, y=kx2+b или y = k sin(x+b)

Промаркировать точки с одной стороны линии как относящиеся к 1 классу, а с другой стороны – ко второму классу. Запомнить разбиение точек как X-Y2.

2. Создать персептрон с двойным входом.

Задать его весовые коэффициенты и смещение равными случайными числами.
Обучить персептрон на выборке X-Y1 с помощью набора функций train, trainc, learnp.
Определить количество эпох, затраченное для обучения (чтобы ошибка была равна 0) – для различных начальных значений весовых коэффициентов и смещения. Затем усреднить результат.

Визуализировать процесс обучения:

- постройте динамику изменения весовых коэффициентов [w1,w2], [w1,b], [w2,b] с эпохами обучения;
- постройте динамику (эволюцию) прямых, реализуемых персептроном (по одной прямой за некоторое количество эпох, так. чтобы общее число прямых было не более 10).
- постройте график зависимости ошибки от номера эпохи.

Сравнить полученную прямую с идеальной разделяющей прямой.

3. Обучить персептрон на выборке X-Y1 с помощью набора функций train, trainc,

learnpn (обучение с нормализацией). Определить количество эпох обучения, требуемое для достижения нулевой ошибки и сравнить с результатами предыдущего пункта.

4. Обучить персептрон на выборке X-Y1 с применением последовательного случайного алгоритма обучения trainr. персептрон на выборке X-Y1 с применением пакетного алгоритма обучения trainb. Определить количество эпох обучения, требуемое для достижения нулевой ошибки с использованием функций обучения learnp, learnpn. Сравнить результаты с последовательным обучением с использованием функции trainc.

5. Обучить персептрон на выборке X-Y1 с применением пакетного алгоритма обучения trainb. Определить количество эпох обучения, требуемое для достижения нулевой ошибки с использованием функций обучения learnp, learnpn. Сравнить результаты с последовательным обучением с использованием функции trainc, trainr.

6. Для выборки X-Y2 произвести обучение персептрона с использованием последовательных (trainc, trainr) и пакетного (trainb) алгоритмов обучения.

Зам. В данном случае невозможно выполнить обучение без ошибки. Поэтому при обучении необходимо либо задавать ненулевую ошибку, при достижении которой обучение необходимо прекращать (net.trainparam.goal), либо задавать максимальное количество эпох обучения (net.trainparam.epochs).

Задать реально достижимое значение ошибки и определить количество эпох обучения, после которого это значение ошибки достигается.

Сравнить результаты.

Определить среднюю ошибку классификации. Представить графически результаты обучения (разбиение на классы, реализуемое персептроном с учетом реальной принадлежности данных к классам).

Рассмотрим обучение персетрона на разделение линейно-разделимых классов с глобальной функцией обучения *trainc* и функцией обучения весов и смещений *trainp*.

Для начала придумаем функцию разделения плоскости на 2 класса. Пусть это будет следующая прямая:

$$-0.5x-y+0.75=0$$

Она делит квадрат $(0,0)-(1,1)$ на 2 примерно равных класса.

Для цели визуализации процесса обучения была написана программа *task5*. В ней максимальное количество эпох равно 1. Функция обучения вызывается в цикле, чтобы была возможно получать значения весов, смещения и ошибки каждую итерацию. В конце происходит тестирование сети на большой выборке, подсчет ошибки и построение графиков.

*Программа **task5***

```
clear all
MAX_EPOCHS = 20;
train_fun = 'trainc';
learn_fcn = 'learnp';
X_Y1 = rand(1000, 2);
line_fun = @(x, y) .3 * x - .6 * y + .1;
test = get2classes(line_fun, X_Y1);
per = newp([0 1; 0 1], 1);
per.inputWeights{1,1}.initFcn='rands';
per.biases{1}.initFcn='rands';
per= init(per);
per.trainFcn = train_fun;
per.inputWeights.learnFcn = {learn_fcn};
per.biases.learnFcn = {learn_fcn};
per.trainParam.epochs = 1;
per.trainParam.showWindow = 0;
WB = zeros(MAX_EPOCHS, 3); PERF = zeros(MAX_EPOCHS, 1);
epochs = 0;

while true
   [per, f] = train(per, X_Y1', test');
   WB(epochs + 1, :) = getwb(per);
   PERF(epochs + 1) = f.best_perf;
   epochs = epochs + 1;
```

```
    if f.best_perf == 0 , break; end
end

newr = rand(2000, 2);
real = get2classes(line_fun, newr);
newp_res = per(newr');
error = 0;
for i = 1:length(newr)
    if (real(i) ~= newp_res(i)), error = error + 1; end
end
disp(["Accuracy: " (1 - error/length(newr))]);
plot2classes(newr, real);
plot2classes(newr, newp_res');

WB = WB(1:epochs, :); PERF = PERF(1:epochs, :);
Figure % Plot w1 and w2
hold on
plot(1:epochs, WB(:,2), 'r');
plot(1:epochs, WB(:,3), 'b');
legend(["w_{1}" 'w_{2}']);
hold off
Figure % Plot w1 and b
hold on
plot(1:epochs, WB(:,2), 'r');
plot(1:epochs, WB(:,1), 'b');
legend(["w_{1}" "b"]);
hold off
Figure % Plot w2 and b
hold on
plot(1:epochs, WB(:,3), 'r');
plot(1:epochs, WB(:,1), 'b');
legend(["w_{2}" "b"]);
hold off
figure % Plot performance
hold on
plot(1:epochs, PERF, 'r');
legend('perf');
hold off

if epochs < 10
    step = 1;
    last_ind = epochs;
else
    step = fix(epochs / 10);
    last_ind = 9;
end
figure
hold on
for i = 1:last_ind
    curi = (i * step);
    func = @(x) (WB(curi, 2) * x + WB(curi, 1)) / (-WB(curi, 3));
    fplot(func, [0 1]);
end
func = @(x) (WB(epochs, 2) * x + WB(epochs, 1)) / (-WB(epochs, 3));
fplot(func, [0 1], 'k--', 'LineWidth', 1.3);
axis([0 1 0 1])
axis square
hold off
```

Графики зависимости изображены на Рис.5.1.1-5.1.4. Также программа рисует прямые, получаемые на разных эпохах обучения (Рис 5.1.5).

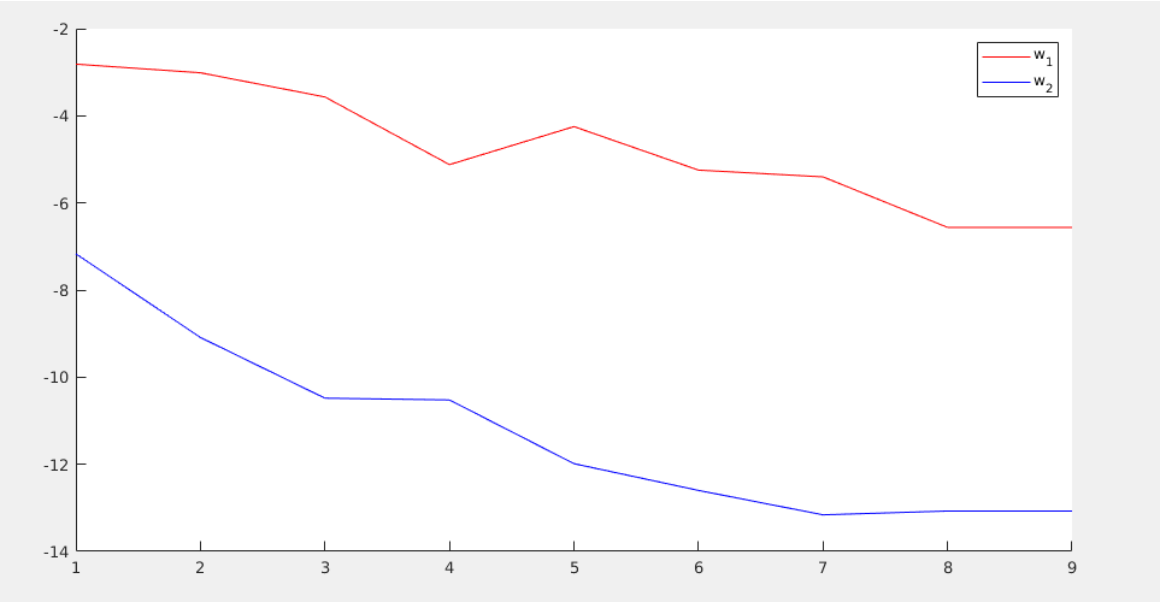

Рис 5.1.1. График зависимости *w1* и *w2* от номера эпохи.

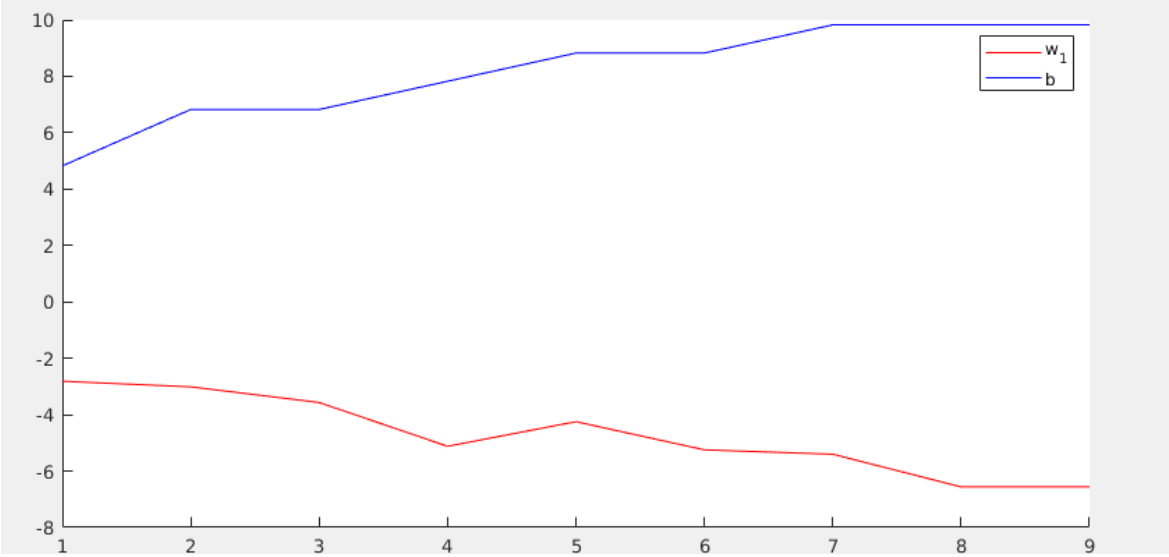

Рис 5.1.2. График зависимости *w1* и *b* от номера эпохи.

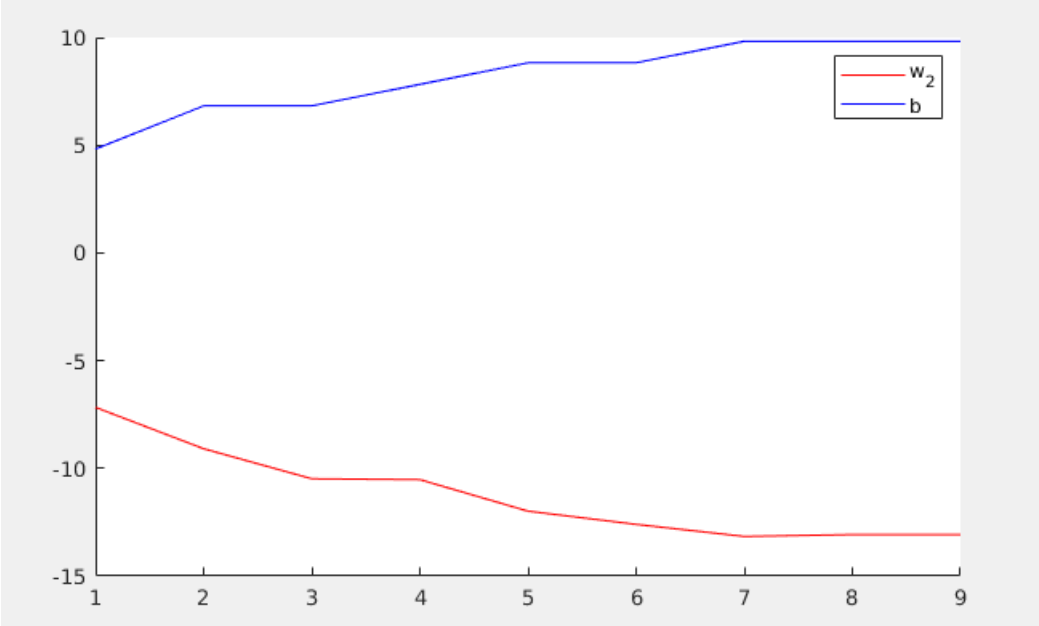

Рис 5.1.3. График зависимости *w2* и *b* от номера эпохи.

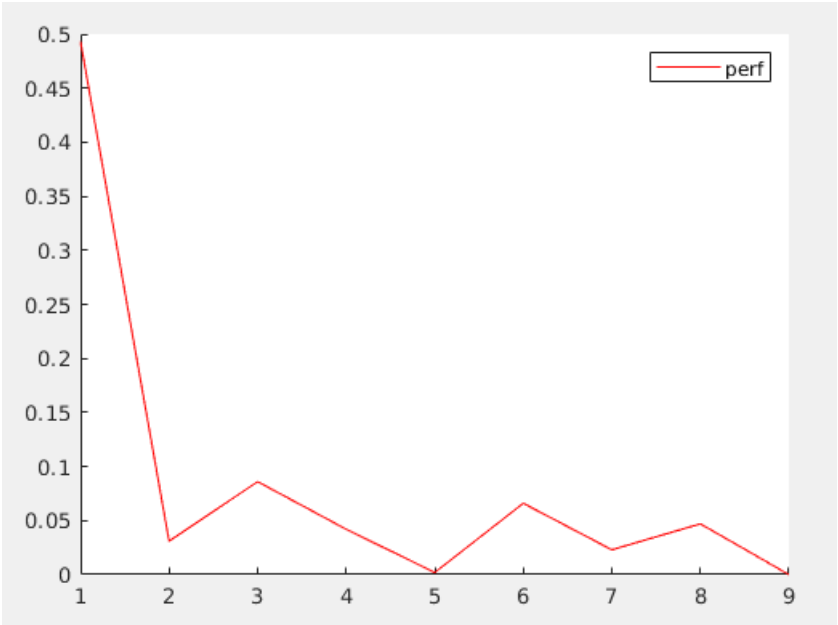

Рис 5.1.4. График зависимости ошибки от номера эпохи.

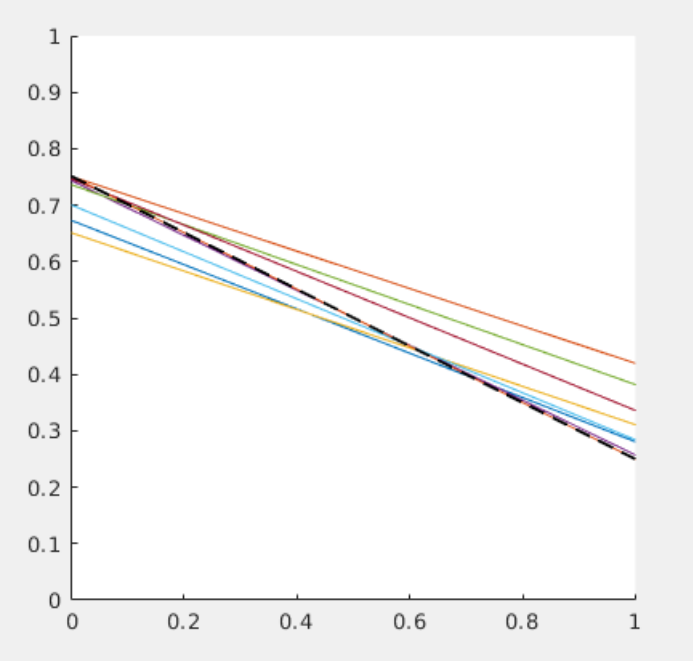

Рис 5.1.5. Прямые на разных эпохах.

Программа *task5* также рисует два графика, на одном правильная классификация на другом классивфикация, полученная персептроном. Классификации изображены на Рис 5.1.6 и Рис 5.1.7 соответсвенно.

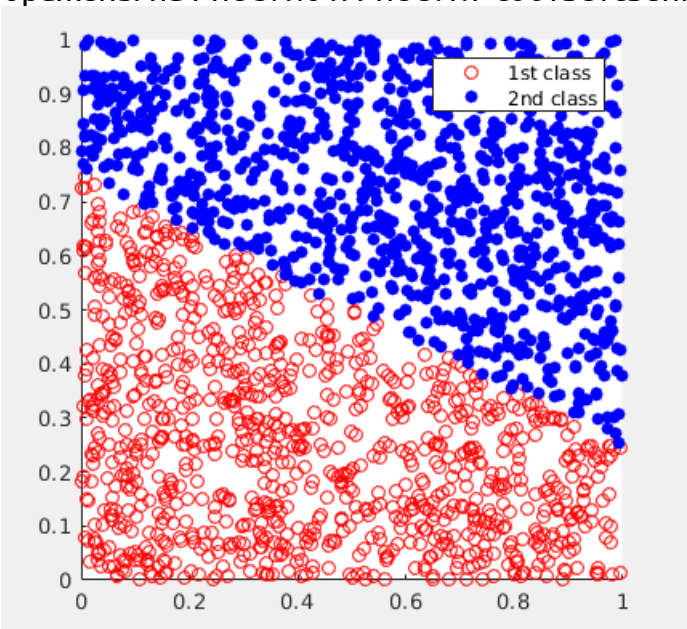

Рис 5.1.6. Правильная классификация.

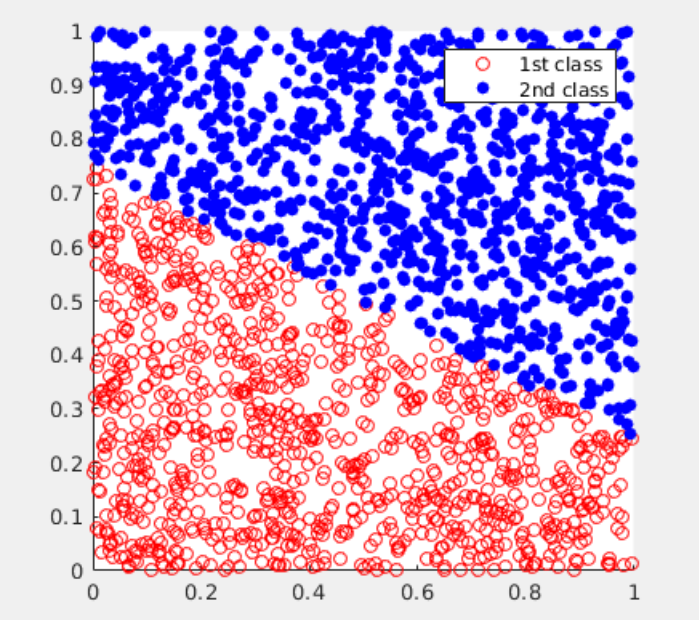

Рис 5.1.7. Классификация, полученная персептроном.

Точность классификации, полученная на тестовой выборке, составила $Accuracy = 0.997$.

6. Вывод

В данной лабораторной было построено вручную несколько нейронных сетей для разделения линейно-неразделимых классов, были проанализированы попытки обучить персептроны для работы с линейно-неразделимыми образами, а также обучен персептрон для распознавания линейно-разделимых классов.