САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ПЕТРА ВЕЛИКОГО

Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет

Лабораторная 3

Исследование нейронных сетей с радиально-базисными функциями (РБФ)

Дисциплина
«Нейроинформатика»

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

Санкт-Петербург
2018

Оглавление

Задание 1. Аппроксимация с помощью РБФ-НС и GRNN.....3

Пункт 1. С помощью точной РБФ-НС (newrbe) аппроксимируйте функцию (для вашего варианта из заданий с номером 5).....3

Пункт 2. Произведите аппроксимацию с помощью, приближенной РБФ-НС (newrb).....6

Пункт 3. Выполните аппроксимацию с помощью GRNN.....10

Пункт 4. Сранение РБФ сетей с сетями ПР.....13

Задание 2. Классификация с помощью PNN (2 класса).....15

Задание 3. Классификация с помощью PNN (>2 классов).....21

Задание 1. Аппроксимация с помощью РБФ-НС и GRNN

Пункт 1. С помощью точной РБФ-НС (newrbe) аппроксимируйте функцию (для вашего варианта из заданий с номером 5)

Задание

а. Задайте достаточное для точной аппроксимации количество обучающих примеров (в данном случае совпадающее с числом РБФ-нейронов).

б. Изменяя ширину РБФ (spread), определите наилучшее значение этого параметра с точки зрения качества аппроксимации. Постройте три графика аппроксимации для разных значений spread: оптимального, больше и меньше оптимального, когда явно видно, что аппроксимация плохая. При построении графиков аппроксимации не забывайте приводить графики исходной желаемой зависимости.

в. Постройте график зависимости ошибки аппроксимации от параметра spread в логарифмическом масштабе.

Напишем программу *task1/task1.m*. В ней создадим выборки из случайных точек, найдем значения функций для этих точек. Затем найдем оптимальное значение spread, которое дает хорошую аппроксимацию.

Программа ***task1.m***

```
%% Prepare data
Ntrain = 1000; % Size of train data
Ntest = 2000; % Size of test data
raw_train = rand(1, Ntrain); % Generate random values for train data
raw_train = cat(2, -raw_train(1:Ntrain/2), raw_train(Ntrain/2 + 1:end) );
raw_test = rand(1, Ntest);
raw_test = sort(cat(2, -raw_test(1:Ntest/2), raw_test(Ntest/2 + 1:end)));
target_train = my_fun(raw_train);
target_test = my_fun(raw_test);
spread = 0.2;
less_spread = 0.001;
more_spread = 0.5;
net_optimal = newrbe(raw_train, target_train, spread);
view(net_optimal);
answer_optimal = net_optimal(raw_test);
net_less = newrbe(raw_train, target_train, less_spread);
answer_less = net_less(raw_test);
net_more = newrbe(raw_train, target_train, more_spread);
answer_more = net_more(raw_test);
fprintf('sse for optimal spread: %f\n', sse(target_test, answer_optimal));
fprintf('sse for less spread: %f\n', sse(target_test, answer_less));
fprintf('sse for more spread: %f\n', sse(target_test, answer_more));
hold on
pl1 = plot(raw_test, target_test, 'xr');
sc1 = plot(raw_test, answer_optimal, 'k','LineWidth', 2);
sc2 = plot(raw_test, answer_less, 'g');
sc3 = plot(raw_test, answer_more, 'b');
legend([pl1 sc1 sc2 sc3], 'Real', 'NN answer (optimal spread)', 'NN answer (less)', 'NN answer (more)');
hold off
```

Ширина РБФ *spread* равная 0.2 дает наилучшию аппроксимацию. Аппроксимации с меньшим, большим и оптимальным значениями приведены на Рис. 1.1.1.

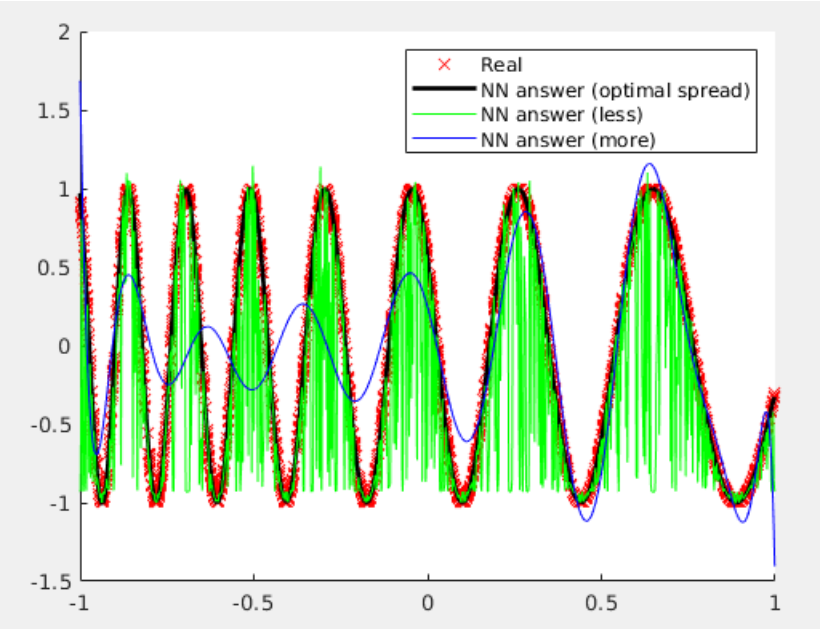

Рис. 1.1.1. Аппроксимация РБФ-НС.

Ошибки (*sse*) приведены в Табл. 1.1.1.

Таблица 1.1.1. Ошибки при различных значениях *spread*.

spread	sse
0.001	706.253347
0.2	0.025850
0.5	465.437803

Посмотрим, как ошибка зависит от значения *spread*. Для этого была написана программа *plot_spread_and_err*. Она принимает на вход три параметра, начальное значение *spread*, шаг и конечное значение. В конце прогрмамма чертик график зависимости ошибки *sse* от значения *spread* в логарифмическом масштабе. Полученный в итоге график представлен на Рис. 1.1.2.

Программа ***plot_spread_and_err***

```
function plot_spread_and_err(start_spread, step_spread, end_spread)
   %% Prepare data
   Ntrain = 1000; % Size of train data
   Ntest = 2000; % Size of test data
   raw_train = rand(1, Ntrain); % Generate random values for train data
   % Make half data negative
   raw_train = cat(2, -raw_train(1:Ntrain/2), raw_train(Ntrain/2 + 1:end) );
   raw_test = rand(1, Ntest);
   raw_test = sort(cat(2, -raw_test(1:Ntest/2), raw_test(Ntest/2 + 1:end)));
   % Creating target data
   addpath('../../../lab0/programs');
   target_train = my_fun(raw_train);
   target_test = my_fun(raw_test);
   spreads = start_spread:step_spread:end_spread;
   err = zeros(1, length(spreads));
   for i = 1:length(spreads)
      spread = spreads(i);
      net = newrbe(raw_train, target_train, spread);
      answer = net(raw_test);
      err(i) = sse(target_test, answer);
   end
   [~, minind] = min(err);
   hold on
   plot(spreads, err, 'r');
   set(gca, 'YScale', 'log');
   set(gca, 'XTick', sort([spreads(minind), get(gca, 'XTick')]));
   hold off
end
```

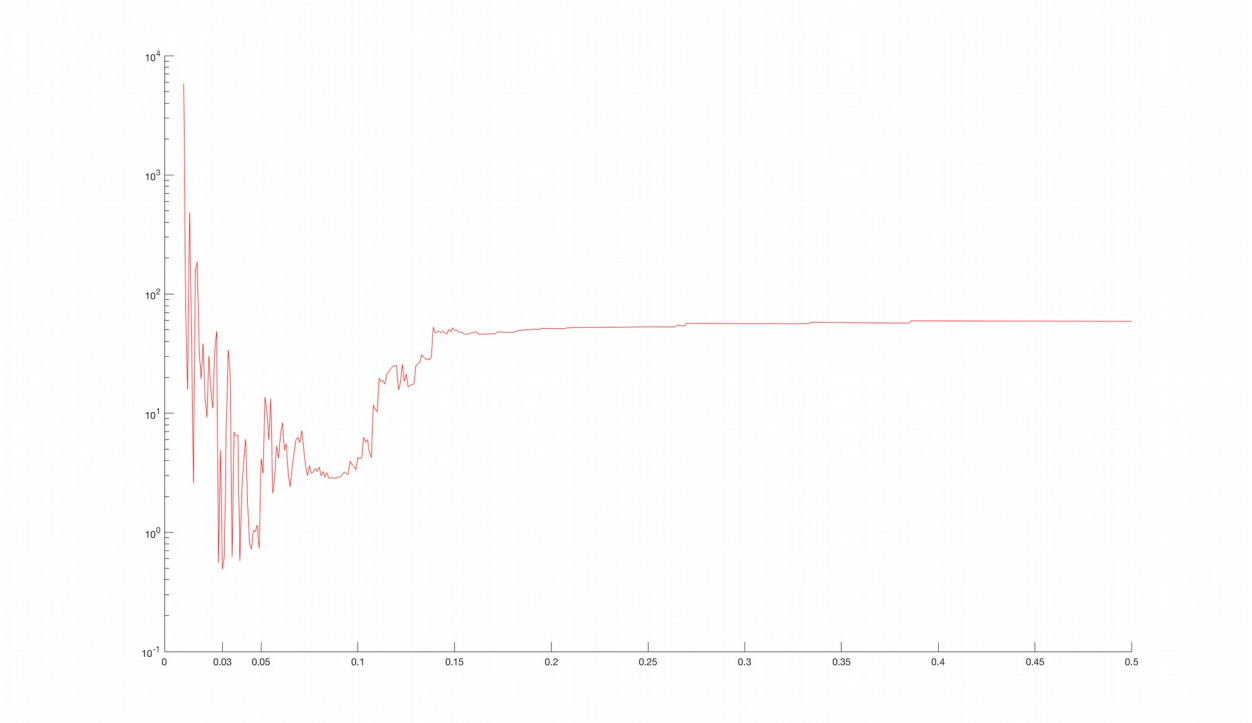

Рис. 1.1.2. График зависимости *sse* от *spread*.

Видим, что минимальное значение достигается при *spread* равным 0.03, после ~0.14 ошибка сильно возрастает, аппроксимация приближается к прямой.

Пункт 2. Произведите аппроксимацию с помощью, приближенной РБФ-НС (newrb)

Задание

а. Используйте найденное в п.1 оптимальное значение spread и достаточный для точной аппроксимации объем обучающей выборки.

б. Изменяя значение допустимой ошибки (goal), постройте зависимость числа используемых РБФ-нейронов от допустимой ошибки аппроксимации. Для промежуточных результатов постройте графики аппроксимации.

Зам. Для более точного нахождения количества нейронов, при которых достигается заданная цель (значение ошибки), количество нейронов, прибавляемое на каждом шаге (5-й параметр newrb, DF), следует задавать не очень большим.

Зам. Максимальное количество нейронов (4-й параметр функции newrb, MN) по построению РБФ-НС не может превышать объема обучающей выборки.

Напишем программу-функцию, которая будет создавать приближенную РБФ-НС с заданной ошибкой аппроксимации. Это программа *task1_2.m*. Параметр df задавался равным единице, чтобы не пропустить значение, при котором была достигнута цель. Ошибка *mse* задавалась от 0.0001 до 0.1, каждый раз увеличивая значение на порядок. Результаты приведены в Табл. 1.2.1. Аппроксимации при различных *goal* показаны на Рис. 1.2.1 – 1.2.4. Программа task1_2_plot.m немного отличается от task1_2, а именно тем, что там еще рисуется полученная аппроксимация. Результаты печатались в консоль.

*Программа **task1_2.m***

```
function neurons = task1_2(goal, df)
  %% Prepare data
  Ntrain = 1000; % Size of train data
  raw_train = rand(1, Ntrain); % Generate random values for train data
  % Make half data negative
  raw_train = cat(2, -raw_train(1:Ntrain/2), raw_train(Ntrain/2 + 1:end) );
  % Creating target data
  target_train = my_fun(raw_train);
  %% Create NN
  spread = 0.03;
  net = newrb(raw_train, target_train, goal, spread, Ntrain, df);
  neurons = net.layers{1}.size;
```

Таблица 1.2.1. Зависимость количества нейронов от ошибки.

goal	neurons
0.0001	65
0.001	77
0.01	34
0.1	31

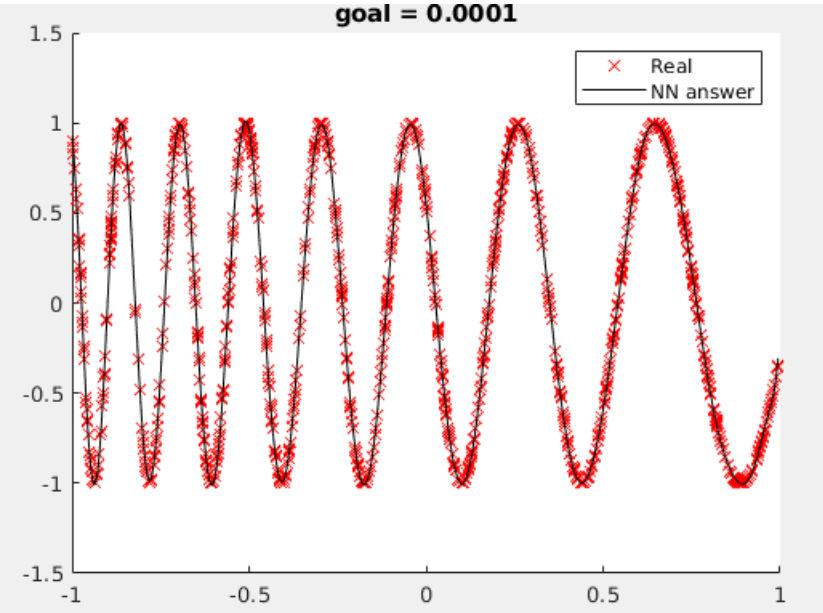

Рис. 1.2.1. Аппроксимация при goal = 0.0001.

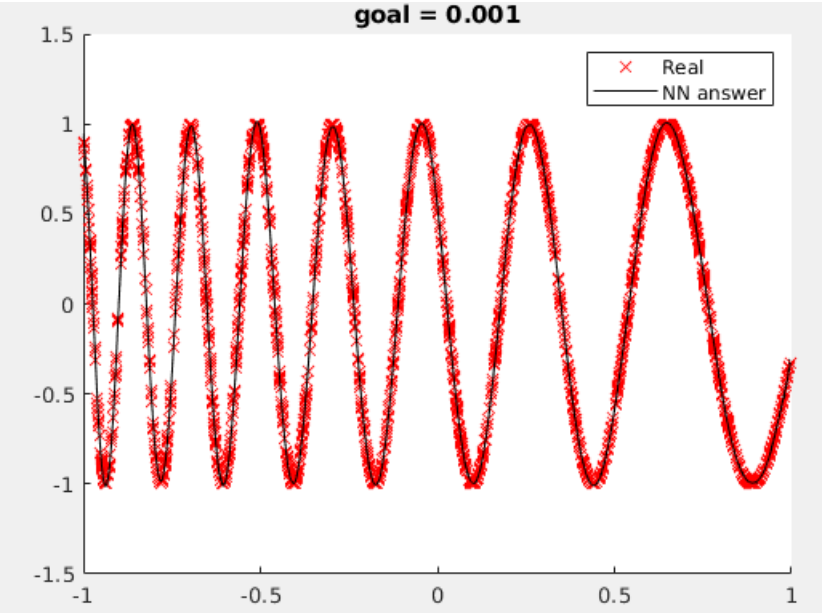

Рис. 1.2.2. Аппроксимация при goal = 0.001.

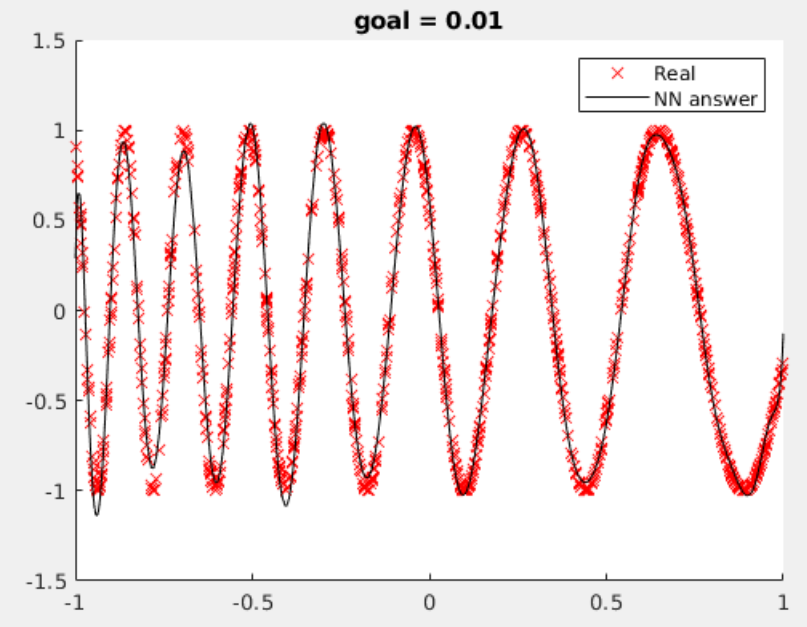

Рис. 1.2.3. Аппроксимация при goal = 0.01.

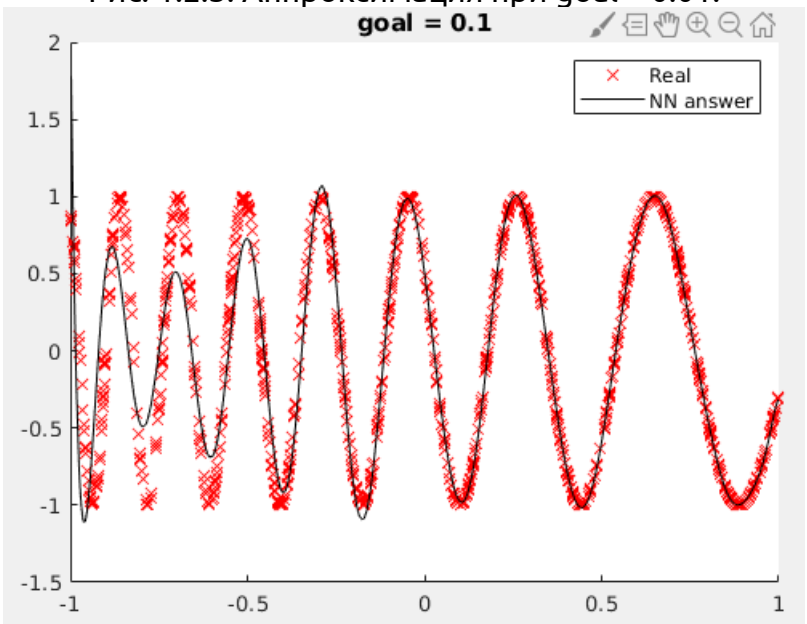

Рис. 1.2.4. Аппроксимация при goal = 0.1.

Для более точного расчета была написана программа *plot_goal_and_neurons*, которая получается на вход стартовую ошибку, шаг и последнюю ошибку. Для каждого значения создается РБФ-НС и и достается количество нейронов в ней. После цикла программа строит график зависимости числа нейронов от цели. Для того, чтобы избежать постоянной печати графика и вывода в консоль были прокомментированы некоторые строчки в файле *newbr.m*. Начальное значение было выбрано 0.0001, шаг = 0.01, последнее значение 0.1001. График изображен на Рис. 1.2.5.

Программа ***plot_goal_and_neurons.m***

```
function plot_goal_and_neurons(start_, step_, end_)

  goals = start_:step_:end_;
  neurons = zeros(1, length(goals));
  ind = 1;
  for goal = goals
    neurons(ind) = task1_2(goal, 1);
    ind = ind + 1;
  end
  close all
  figure; hold on
  plot(goals, neurons, '');
  xlabel('goal');
  ylabel('neurons');
  xlim([0 0.1]);
  hold off
```

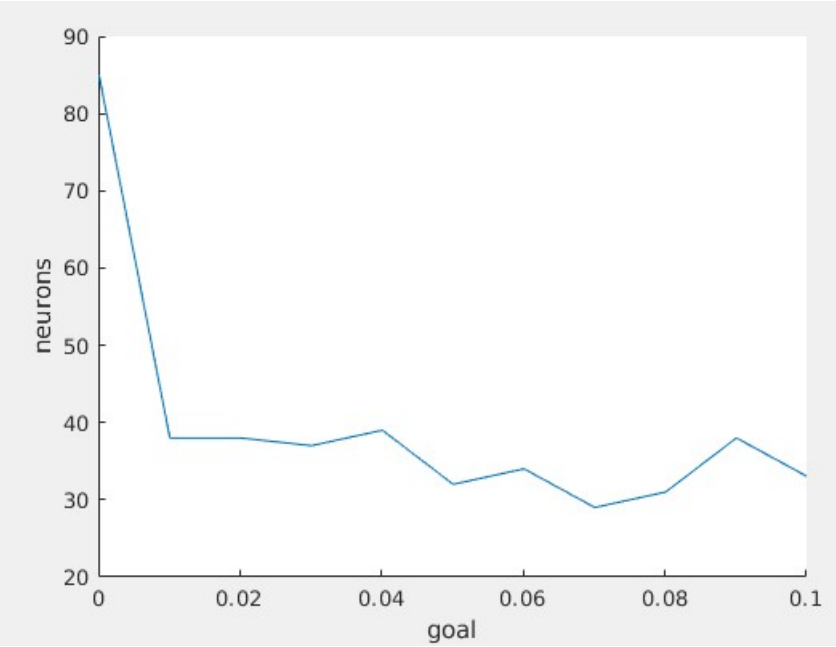

Рис. 1.2.5. График зависимости числа нейронов от цели.

Пункт 3. Выполните аппроксимацию с помощью GRNN.

Задание

а. Задайте вначале достаточный для точной аппроксимации объем обучающей выборки.

б. По аналогии с newrb изменяя ширину РБФ (spread), определите наилучшее значение этого параметра с точки зрения качества аппроксимации. Постройте график аппроксимации на исходной зависимости.

в. Уменьшите объем обучающей выборки в несколько раз (рассмотрите 3 случая) и подберите оптимальные значения параметра spread для каждого случая. Постройте полученные графики аппроксимации. Приведите значения ошибок.

Для аппроксимации с помощью *GRNN* была написана программа *task1_3.m*. В ней мы задаем три тренировочные выборки объемами 1000, 500 и 300. Затем вариьируем значение *spread*, чтобы добиться хорошей аппроксимации.

*Программа **task1_3.m***

```
%% Prepare data
% Testing different train data size
Ntrain1 = 1000; % Size of train data
raw_train1 = rand(1, Ntrain1); % Generate random values for train data
raw_train1 = cat(2, -raw_train1(1:Ntrain1/2), raw_train1(Ntrain1/2 + 1:end) );
Ntrain2 = 500; % Size of train data
raw_train2 = rand(1, Ntrain2); % Generate random values for train data
raw_train2 = cat(2, -raw_train2(1:Ntrain2/2), raw_train2(Ntrain2/2 + 1:end) );
Ntrain3 = 300; % Size of train data
raw_train3 = rand(1, Ntrain3); % Generate random values for train data
raw_train3 = cat(2, -raw_train3(1:Ntrain3/2), raw_train3(Ntrain3/2 + 1:end) );

Ntest = 2000; % Size of test data
raw_test = rand(1, Ntest);
raw_test = sort(cat(2, -raw_test(1:Ntest/2), raw_test(Ntest/2 + 1:end)));
% Creating target data
addpath('../../../lab0/programs');
target_train1 = my_fun(raw_train1);
target_train2 = my_fun(raw_train2);
target_train3 = my_fun(raw_train3);
target_test = my_fun(raw_test);
spread = 0.0006;

net1 = newgrnn(raw_train1, target_train1, spread);
answer1 = net1(raw_test);
net2 = newrbe(raw_train2, target_train2, spread);
answer2 = net2(raw_test);
net3 = newrbe(raw_train3, target_train3, spread);
answer3 = net3(raw_test);
fprintf('sse for 1000 neurons: %f\n', sse(target_test, answer1));
fprintf('sse for 500 neurons: %f\n', sse(target_test, answer2));
fprintf('sse for 300 neurons: %f\n', sse(target_test, answer3));
hold on
title(['Spread = ', num2str(spread)]);
pl1 = plot(raw_test, target_test, 'xr');
sc1 = plot(raw_test, answer1, 'k','LineWidth', 2);
sc2 = plot(raw_test, answer2, 'g');
sc3 = plot(raw_test, answer3, 'b');
legend([pl1 sc1 sc2 sc3], 'Real', 'NN answer (1000 neurons)', 'NN answer (500 neurons)', 'NN answer (300
neurons)');
axis([-1 1 -1 1]);
hold off
```

Результаты для различных значений *spread* и объемов выборки показаны в Табл. 1.3.1. Аппроксимации при трех разных значениях *spread* приведены на Рис. 1.3.1 – 1.3.3.

Таблица 1.3.1. Ошибки при различных spread и neurons.

spread	neurons	sse
0.0006	1000	1.168559
	500	1708.829768
	300	1745.337087
0.005	1000	0.880716
	500	78.796100
	300	293.970787
0.2	1000	910.169204
	500	0.021850
	300	0.088819

При слишком малом количестве нейронов трудно подобрать значение *spread*, так как нейронов недостаточно для хорошей аппроксимации. Мы видим, что чем меньше нейронов мы назначаем, тем больше нужно задавать *spread*.

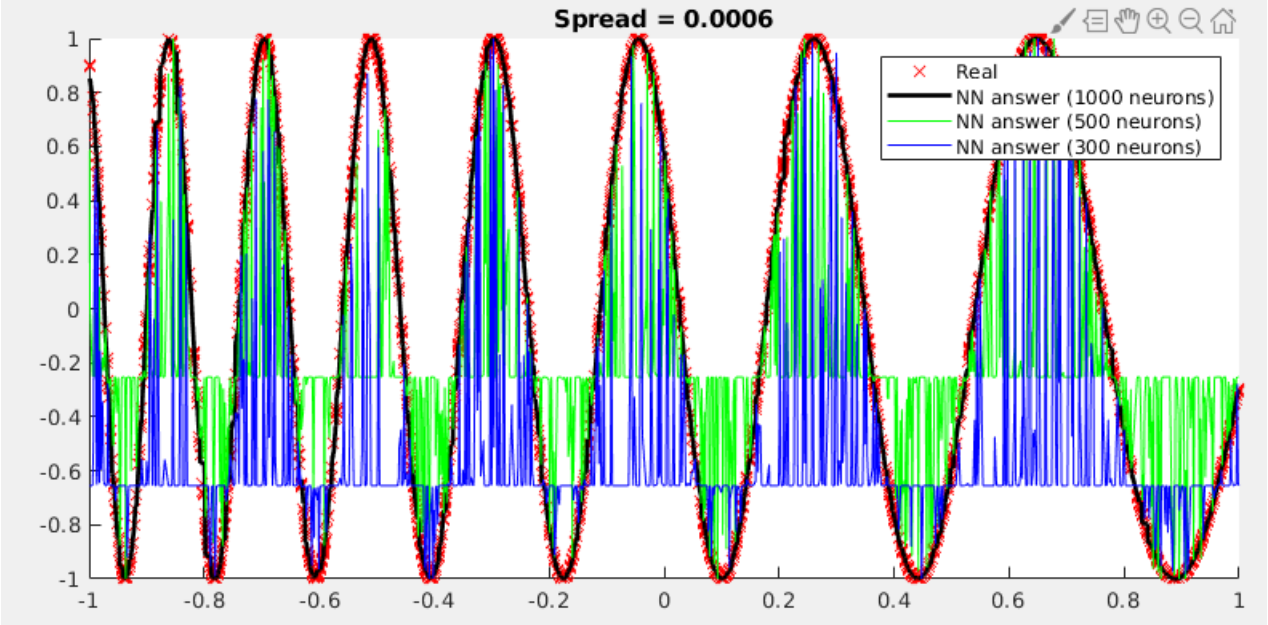

Рис. 1.3.1. Аппроксимация *GRNN* при *spread* = 0.0006.

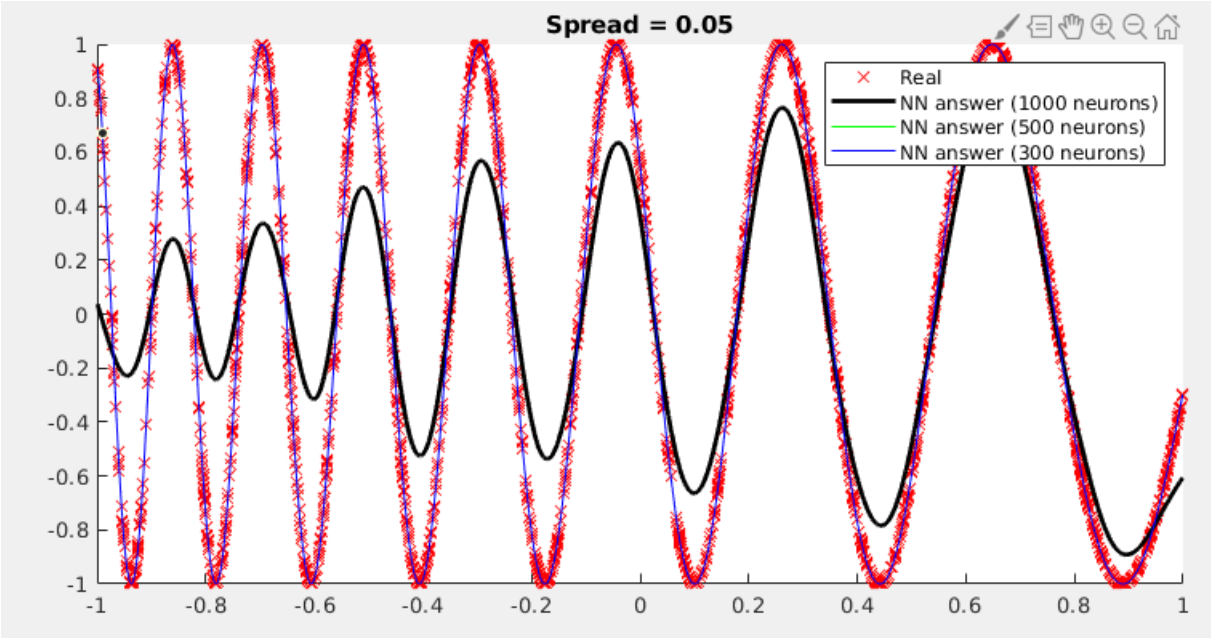

Рис. 1.3.2. Аппроксимация *GRNN* при *spread* = 0.005.

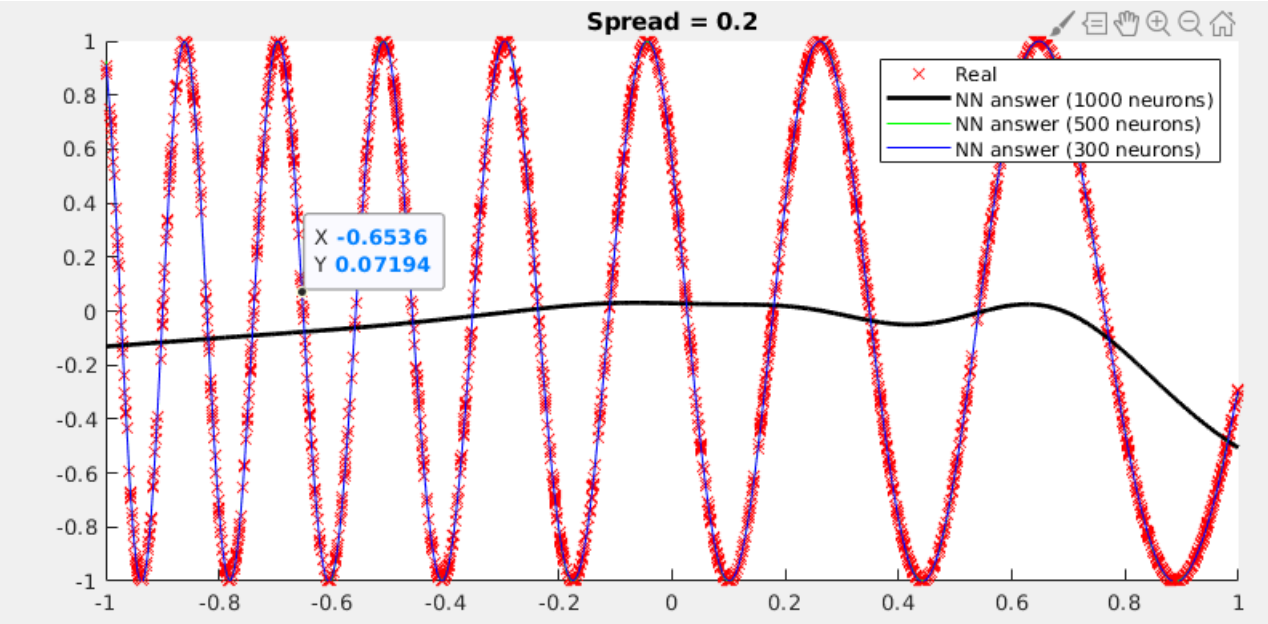

Рис. 1.3.3. Аппроксимация *GRNN* при *spread* = 0.2

Пункт 4. Сранение РБФ сетей с сетями ПР

Задание

Сравните качество аппроксимации сетями прямого распространения и различными вариантами РБФ-НС (точной, приближенной РБФ-НС и GRNN) по различным показателям:

- число нейронов, требуемое для достижения заданного качества аппроксимации;
- время обучения;
- сложность настройки (выбора параметров) НС.

Постройте на одном графике зависимости ошибки от числа нейронов для различных типов НС.

Зам. При расчете ошибок для различных типов НС следует использовать одни и те же формулы.

Сравним РБФ сети с сетями прямого распространения, для этого возмем сеть из предыдущей лабораторной, которая наилучшим образом аппроксимировала функцию. Для сетей ПР не будем заново ничего строить, так как результаты все уже есть. Они будут взяты из lab2.

Если сравнивать по количеству нейронов, то сети прямого распространения, очевидно, оказываются безусловными победителями, так как дают хорошую аппроксимацию уже при 52 нейронах для однослойной сети и при 35 нейронах при трехслойной (20 10 5), эти показатели давал алгоритм обучения *trainlm*, *sse* при этом, для трехслойной сети достигал значения 0.05.

Если говорить о времени обучения, то выигрывают здесь точная РБФ-НС и GRNN, так как обучаются они практически мгновенно. Сети же прямого распространения могут требовать достаточно много времени на обучение.

Самыми простыми в настройке являются точные РБФ-НС, для их настройки нужно указывать лишь два параметра: размер выборки – количество нейронов и *spread*. Некоторые сети ПР бывают очень капризными, и их настройка может занимать достаточно много времени.

Строить зависимости ошибки от числа нейронов для НСПР и РБФ-ПР не считаю правильным, так как очевидно, что РБФ-НС требуют намного большего количества нейронов, если пытаться обучить НСПР с таким же количеством нейронов, с какими обучаются РБФ-НС, то можно будет ждать целую вечность. Поэтому я взял график зависимости ошибки НСПР из предыдущей лабораторной. А для построения зависимостей для точной РБФ-НС написал программу *task1_4*. Для приближенной РБФ-НС не будем строить такой график, так как эти сети строятся по другому принципу. И для GRNN тоже не будет строить график, так как эта сеть для разного количества нейронов требудет разное значение *spread*, в чем мы убедились в предыдущем пункте. В итоге построим график зависимости ошибки от числа нейронов только для точной РБФ-НС.

*Программа **task1_4.m***

```
%% Prepare data
Ntest = 2000; % Size of test data
raw_test = rand(1, Ntest);
raw_test = sort(cat(2, -raw_test(1:Ntest/2), raw_test(Ntest/2 + 1:end)));
% Creating target data
```

```
addpath('../../../lab0/programs');
target_test = my_fun(raw_test);

%% Create network
spread = 0.03;
min_n = 10;
delta = 10;
max_n = 2001;

neurons = min_n:delta:max_n;
sses = zeros(1, length(neurons));

for i = 1:length(neurons)
    Ntrain = neurons(i); % Size of train data and num of neurons in rb
    raw_train = rand(1, Ntrain); % Generate random values for train data
    raw_train = cat(2, -raw_train(1:Ntrain/2), raw_train(Ntrain/2 + 1:end));
    target_train = my_fun(raw_train);
    net = newrbe(raw_train, target_train, spread);
    answer = net(raw_test);
    sses(i) = sse(target_test, answer);
end

hold on
plot(neurons, sses);
xlabel('neurons');
ylabel('sse');
set(gca, 'YScale', 'log');
hold off
```

Характеристика из предыдущей лабораторной для НСПР представлена на Рис. 1.4.2. Характеристика для РБФ-НС – на Рис. 1.4.1.

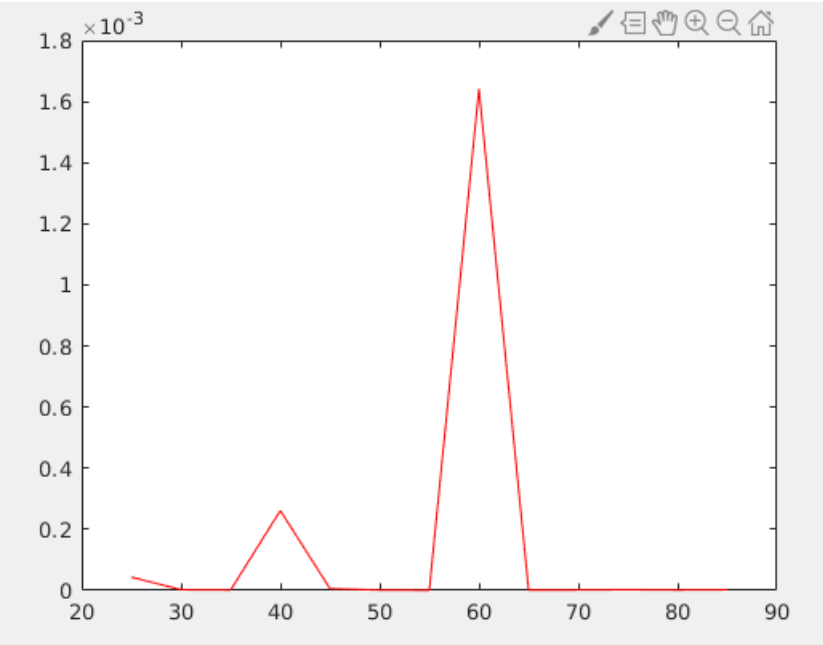

Рис. 1.4.1. Зависимость sse от количества нейронов для НСПР.

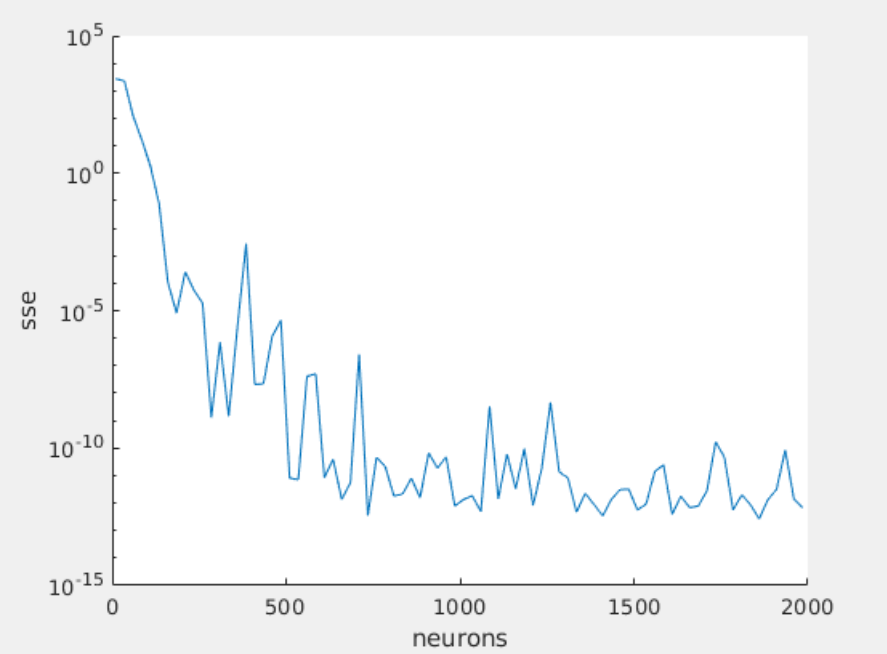

Рис. 1.4.2. Зависимость sse от количества нейронов для РБФ-НС.

Видим, что графики совершенно разные. Для РБФ-НС нормальная ошибка получается только при количестве нейронов более 200. Далее, после 500 ошибка перестает уменьшаться, только иногда присутствуют пики. Для НСПР же брались средние значения после 10 запусков, однако кривая все равно получилась совершенно не гладкой.

Задание 2. Классификация с помощью PNN (2 класса)

Задание

1. Задайте достаточное для точной классификации количество обучающих примеров.
2. Подберите оптимальное значение spread в смысле минимальной ошибки на тестовой выборке. Визуализируйте результаты классификации (диаграмма соотнесения тестовых примеров с классами, раскраска плоскости). Приведите значение средней ошибки.
3. Постройте дополнительно графики классификации для значений spread, больших и меньших оптимального, когда явно видно, что классификация неудовлетворительная.
4. Уменьшите объем обучающей выборки в несколько раз (рассмотрите 3 случая) и подберите оптимальные значения параметра spread для каждого случая. Визуализируйте результаты классификации и приведите значение средней ошибки.
5. Постройте поверхность ошибки в плоскости двух параметров: ширина РБФ- функции spread и объем обучающей выборки.
6. Сравните полученные результаты с НС прямого распространения по аналогии с п. 3 задания 1.

Найдем для начала подходящее значение *spread* и подберем объем выборки такой, чтобы получить хорошую классификацию. Программа *task2* принимает на

вход 3 параметра: значение *spread*, объем обучающей выборки и логическое значение, определяющее нужно ли визуализировать результат.

Программа ***task2***

```
function [cross, mse_] = task2(spread, Ntrain, pl)
    % Preparing
    Ntest = 4000;
    raw_train = rand(2, Ntrain);
    raw_test = rand(2, Ntest);
    % is_in_area from lab0/programs/
    addpath('../../../lab0/programs');
    Ttrain = is_in_area(raw_train');
    Ttest = is_in_area(raw_test');
    t = [Ttrain; ~Ttrain];

    net = newpnn(raw_train, t, spread);
    answer = net(raw_test);
    cross = crossentropy([Ttest; ~Ttest]', answer');
    mse_ = mse([Ttest; ~Ttest]', answer');
    if pl; plot2classes(raw_test', answer'); end
```

Выяснили, что для аппроксимации исходных классов подходят следующие параметры: *spread = 0.0068*, и *Ntrain = 10000*. Аппроксимации, полученные при разных значениях показаны на Рис. 2.1 – 2.3. Все полученные далее результаты для различных *spread* занесены в Табл. 2.1.

Таблица 2.1. Ошибки при разных spread.

spread	Ntrain	mse	crossentropy
0.0069	10000	0.0118	0.1262
0.1		0.1037	1.2120
0.0001		4.0940	9.5831

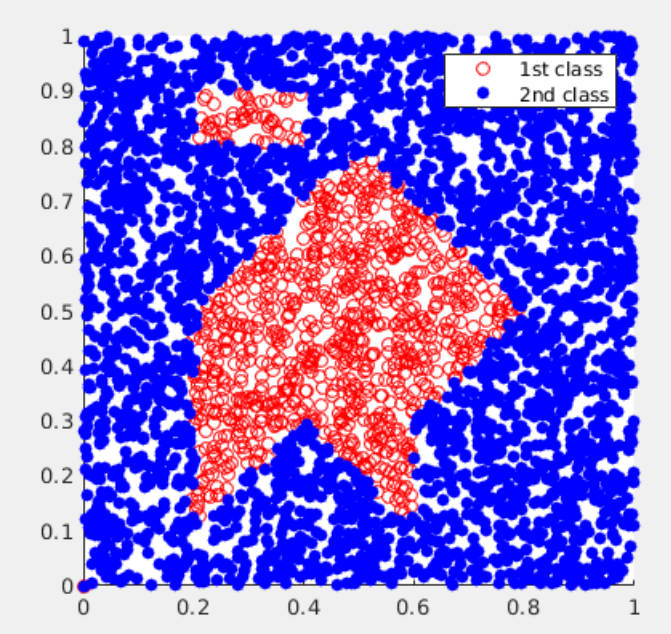

Рис. 2.1. Аппроксимация PNN. Spread = 0.0069, Ntrain = 10000.

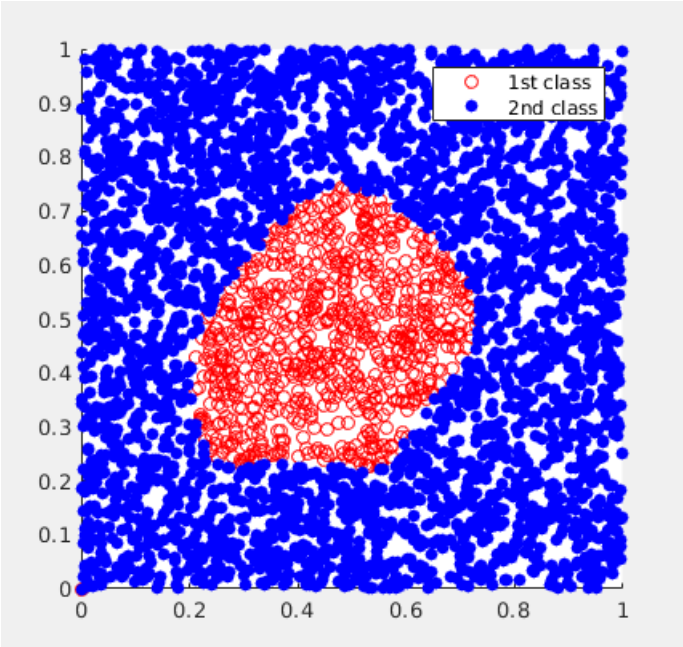

Рис. 2.2. Аппроксимация PNN. Spread = 0.0001, Ntrain = 10000.

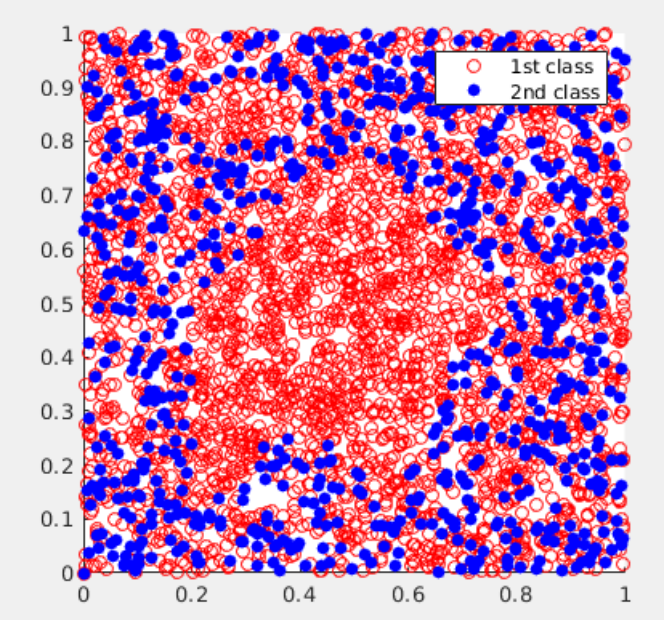

Рис. 2.3. Аппроксимация PNN. Spread = 0.1, Ntrain = 10000.

Попробуем теперь уменьшить выборку в несколько раз и подобрать spread. Значения ошибок представлены в Табл. 2.2. Аппроксимации изображены на Рис. 2.4 – 2.7.

Таблица 2.2. Ошибки при разных spread и Ntrain.

spread	Ntrain	mse	crossentropy
0.0069	1000	0.0440	0.5632
0.0007	500	0.0697	0.7885
0.007	300	0.0805	0.8560

При сильном уменьшее количества нейронов аппроксимация становится слишком плохой, как бы не менялось значение *spread*.

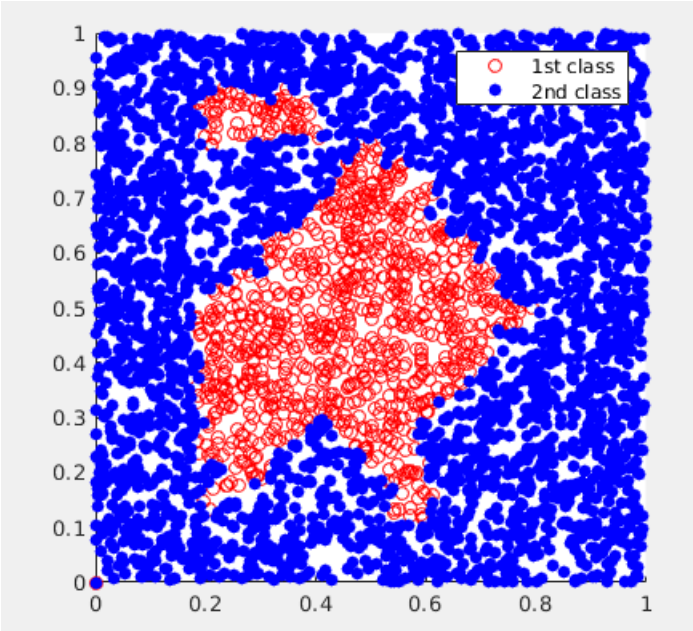

Рис. 2.4. Аппроксимация PNN. Spread = 0.0069, Ntrain = 1000.

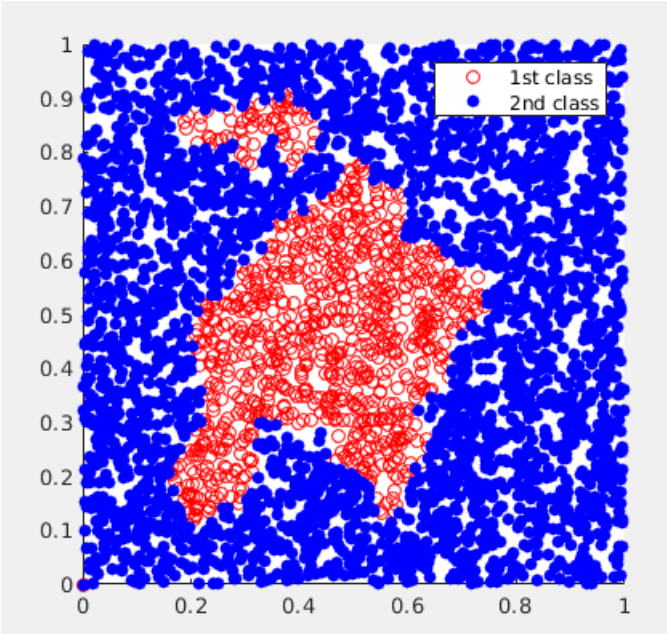

Рис. 2.5. Аппроксимация PNN. Spread = 0.0007, Ntrain = 500.

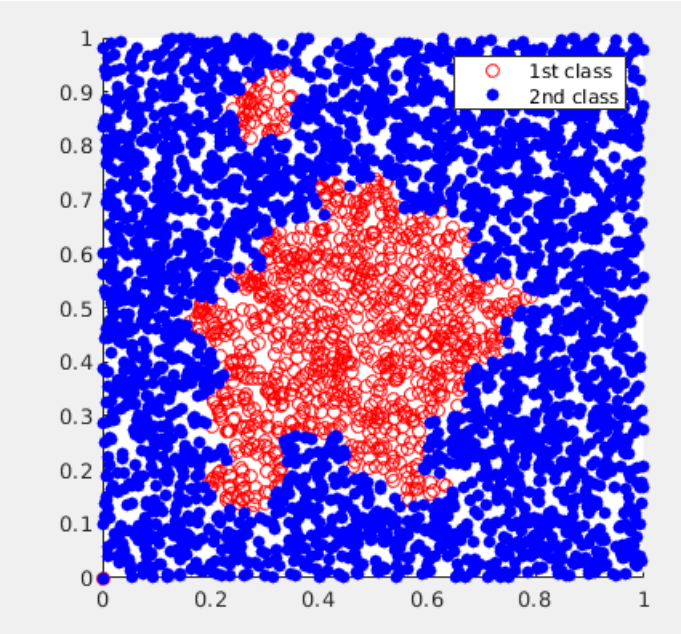

Рис. 2.6. Аппроксимация PNN. Spread = 0.007, Ntrain = 300.

Напишем программу, которая будет запускать функцию *task2* в цикле, меняя значение *Ntrain* и *spread*. Программа будет сохрянять полученные ошибки и строить поверхность ошибок.

Программа ***errsurf***

```
function surface = errsurf(start_n, delta_n, end_n, start_s, delta_s, end_s)
    neurons = start_n:delta_n:end_n;
    spreads = start_s:delta_s:end_s;
    errs = zeros(length(neurons), length(spreads));
    for i = 1:length(neurons)
        for j = 1:length(spreads)
            [cross, ~] = task2(spreads(j), neurons(i), 0);
            errs(i, j) = cross;
        end
    end
    surface = errs;
    figure
    surf(neurons, spreads, errs,'FaceAlpha',0.8);
    colorbar
    xlabel('neurons');
    ylabel('spread');
    zlabel('crossentropy');
    hold on
    s = size(errs);
    z = zeros(s);
    surf(neurons, spreads, z, errs);
    xlim([start_n end_n]);
    ylim([start_s end_s]);
    view(127.5, 30)
```

Построили поверхность ошибок для *Ntrain* от 100 до 10100 с шагом 500 и для *spread* от 0.0001 до 0.0101 с шагом 0.0005. Полученный результат приведен на Рис. 2.7.

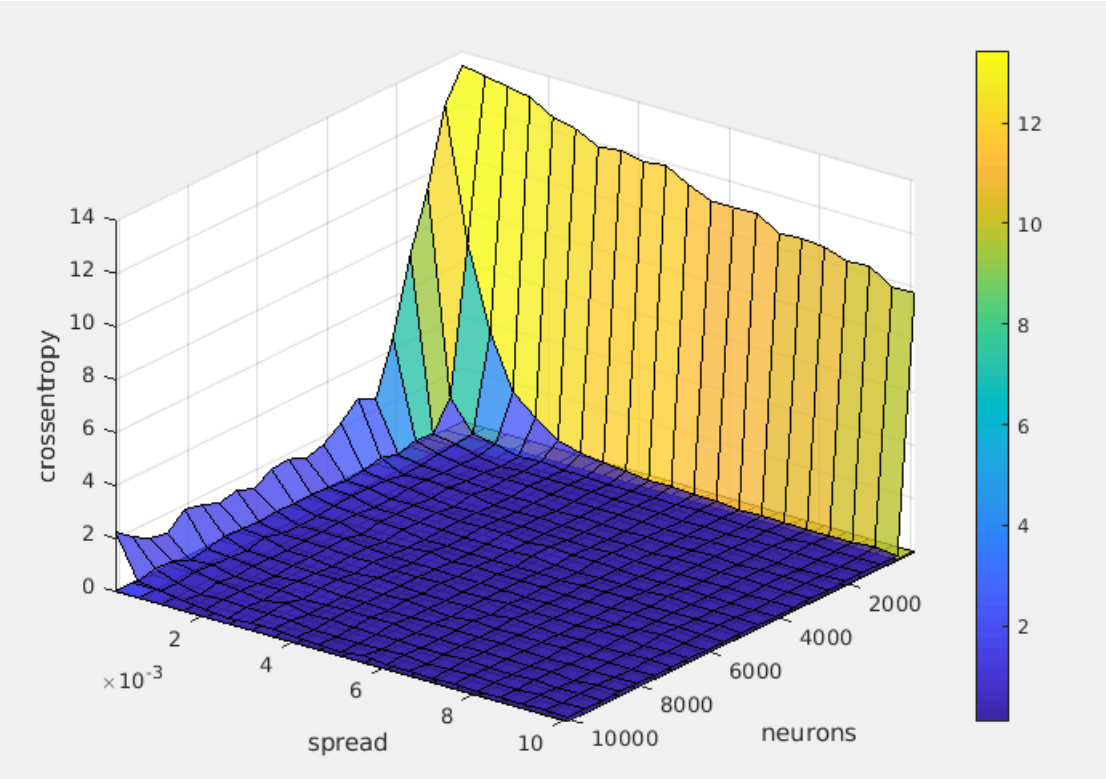

Рис. 2.7. Зависимость ошибки от числа нейронов и значения *spread*.

Видно, что чем больше нейронов, тем меньше становится ошибка. Значения *spread* в заданном диапазоне дают маленькую ошибку.

Если сравнивать PNN с НСПР, то по точности аппроксимации они примерно похожи, по показателю *crossentropy* сравнивать неправильно, потому что в предыдущей лабораторной выходной слой был *softmax*, а здесь compet, который на выходе сразу дает строго либо 0 либо 1, следовательно и ошибка crossentropy будет выше.

По времени обучения выигрывает PNN, как и в предыдущем пункте, здесь все происходит практически моментально, в отличии от НСПР.

По сложности настройки опять же выигрывает сеть с РБФ слоем.

Задание 3. Классификация с помощью PNN (>2 классов)

Задание

С помощью PNN произведите классификацию линейно неразделимых образов (для вашего варианта из заданий с номером 4) по аналогии с заданием 2.

Для п. 2 и 3 дополнительно приведите матрицы неточностей и рассчитайте ошибки первого и второго родов.

Напишем программу по аналогии с предыдущим заданием, только для 5 классов.

*Программа **task3.m***

```
function [cross, mse_] = task3(spread, Ntrain, pl)
    % Preparing
    Ntest = 4000;
    raw_train = rand(2, Ntrain);
    raw_test = rand(2, Ntest);
    % is_in_area from lab0/programs/
    addpath('../../../lab0/programs/raw_data4_2');
    Ttrain = is_in_area2(raw_train', 2);
    Ttest = is_in_area2(raw_test', 2);


    net = newpnn(raw_train, Ttrain, spread);
    view(net);
    answer = net(raw_test);
    cross = crossentropy(Ttest, answer)
    mse_ = mse(Ttest, answer)
    if pl
        figure; plot7classes(raw_test', answer);
        figure; plotconfusion(Ttest, answer);
    end
```

Найдем оптимальное значение Ntrain и spread.

Задали *Ntrain* = 10000, *spread* = 0.008. Получили ошибку *crossentropy* = 0.0775, *mse* = 0.0043.

Полученная аппроксимация показана на Рис. 3.1, а матрица неточностей и ошибки 1-го и 2-го родов показаны на Рис. 3.2.

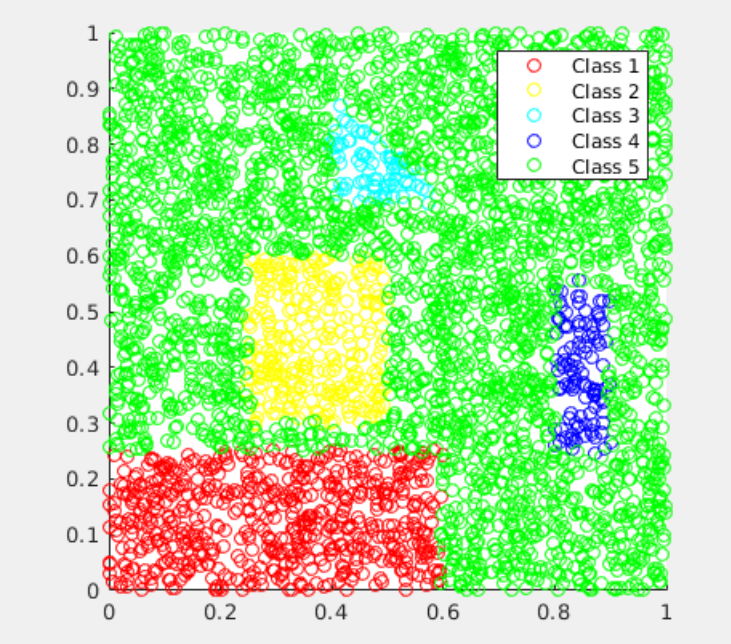

Рис. 3.1. Аппроксимация PNN, spread = 0.008, Ntrain = 10000.

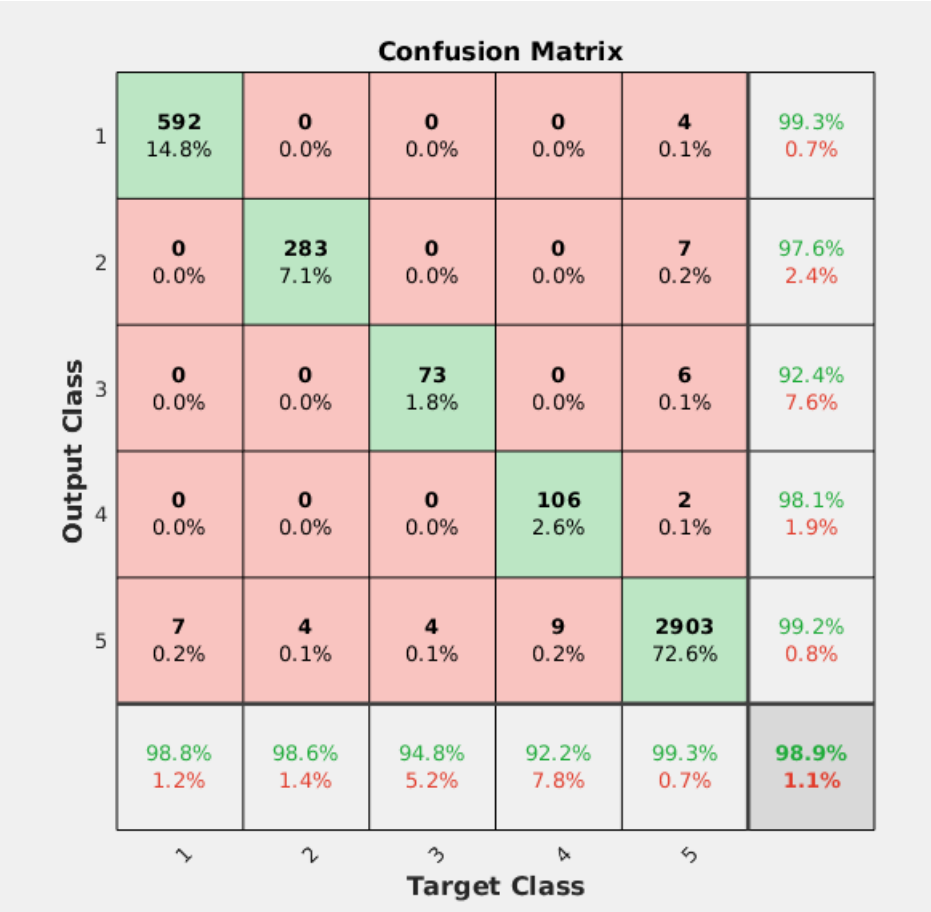

Рис. 3.2. Матрица неточностей, spread = 0.008, Ntrain = 10000.

Поменяем теперь *spread* при неизменном значении *Ntrain*. Попробуем увеличить его и уменьшить. Результаты для большого *spread* показаны на Рис. 3.3. и 3.4. Результаты для маленького *spread* – на Рис. 3.5 и 3.6. Результаты для всех трех значений *spread* показаны в Табл. 3.1.

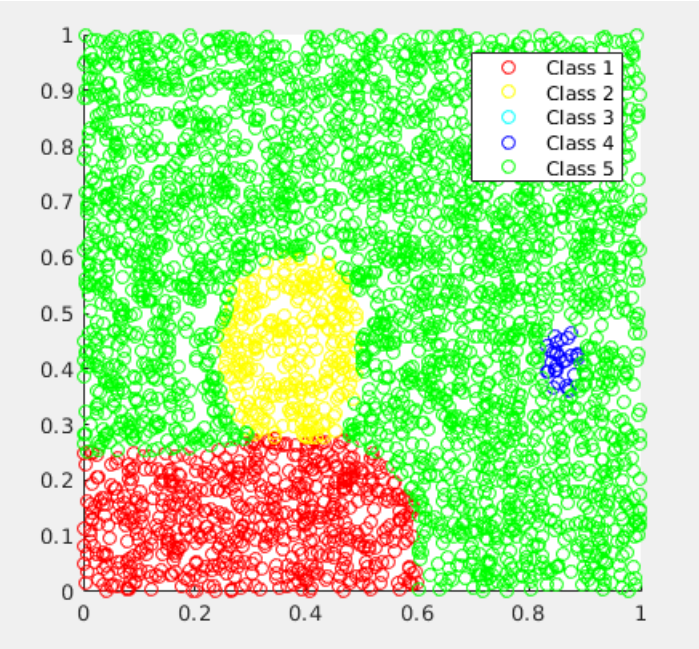

Рис. 3.3. Аппроксимация PNN, spread = 0.08, Ntrain = 10000.

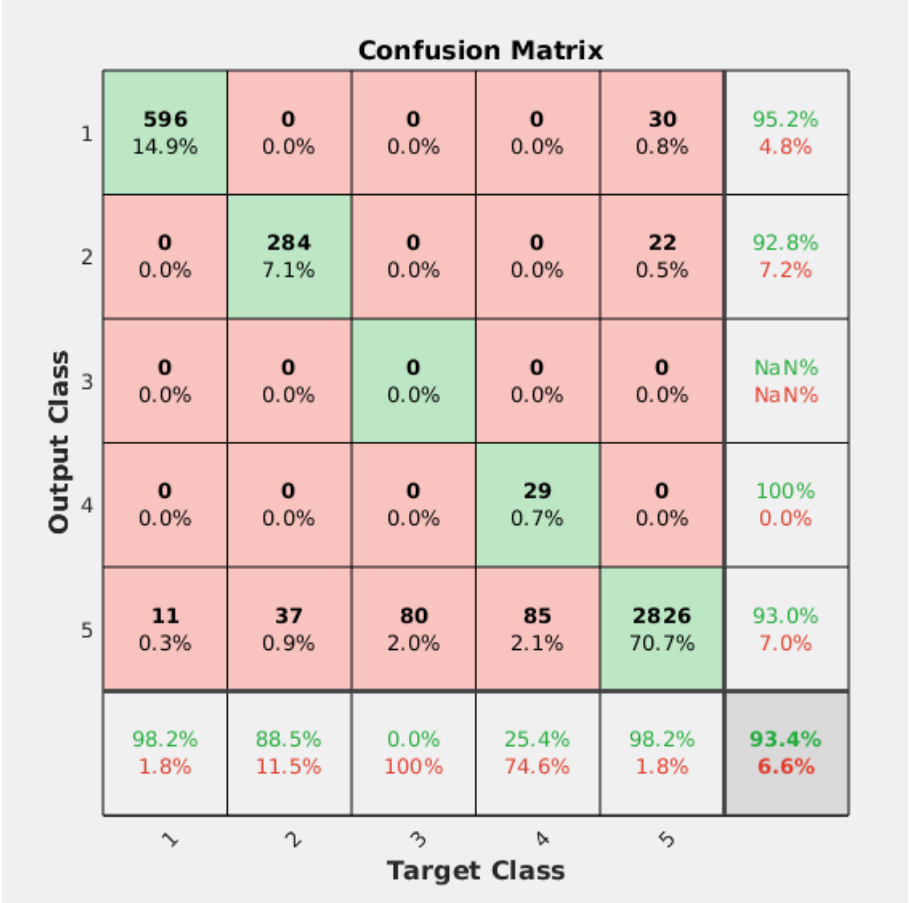

Рис. 3.4. Матрица неточностей, spread = 0.08, Ntrain = 10000.

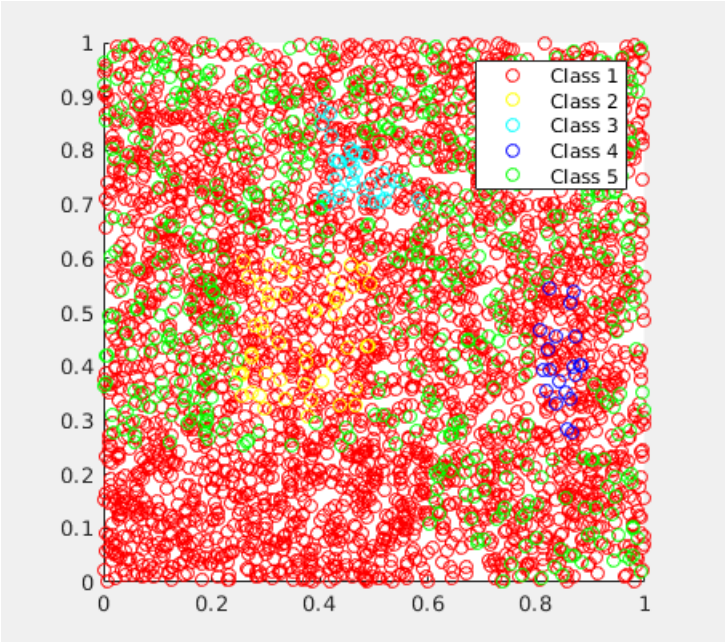

Рис. 3.5. Аппроксимация PNN, spread = 0.00008, Ntrain = 10000.

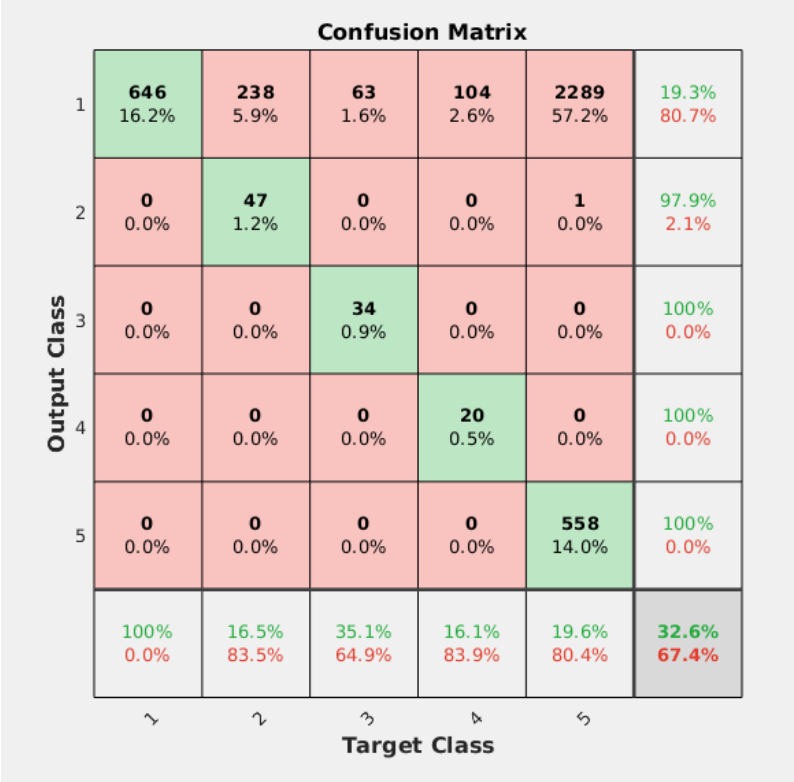

Рис. 3.6. Матрица неточностей, spread = 0.00008, Ntrain = 10000.

Таблица 3.1. Ошибки для разных spread.

spread	Ntrain	crossentropy	mse	accuracy
0.008	10000	0.0775	0.0043	98.9%
0.08		0.4776	0.0265	93.4%
0.00008		4.8569	0.2695	32.6%

При очень маленьком значении *spread* получаем большую ошибку.

Попробуем теперь уменьшить объем выборки в несколько раз и подобрать для них *spread*, чтобы получить нормальную аппроксимацию. Полученные результаты представлены в Табл. 3.1, а также на Рис. 3.7 – 3.10.

Таблица 3.1. Ошибки для разных spread.

spread	Ntrain	crossentropy	mse	accuracy
0.008	2000	0.2054	0.0114	97.2%
0.01	500	0.3352	0.0186	95.3%

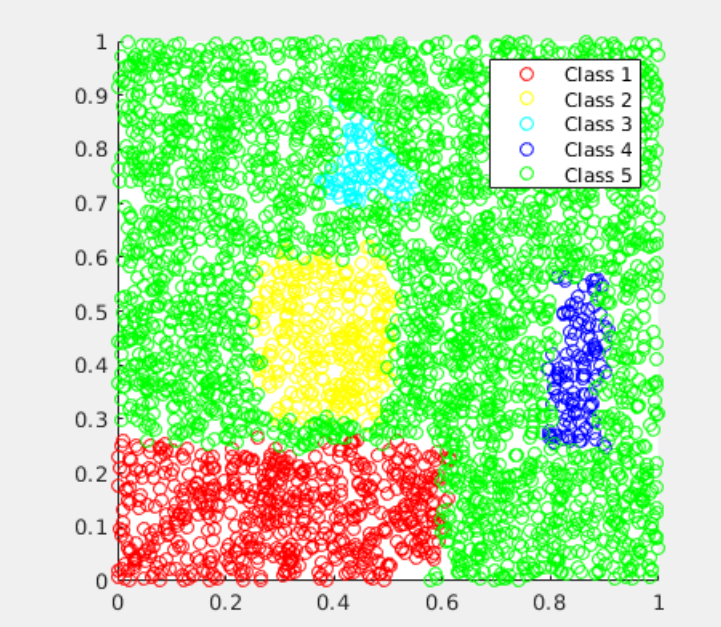

Рис. 3.7. Аппроксимация PNN, spread = 0.008, Ntrain = 2000.

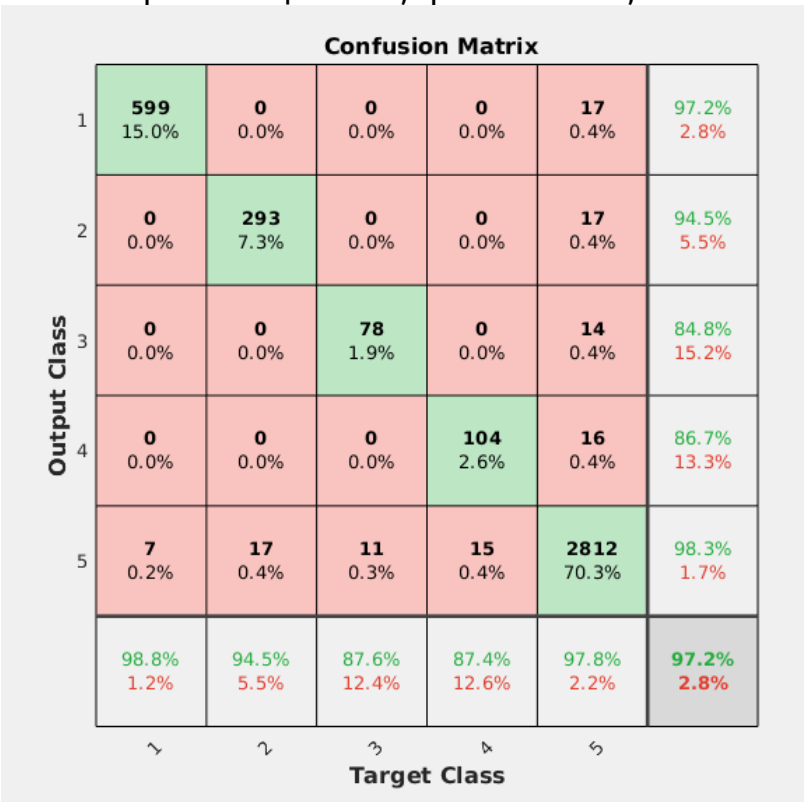

Рис. 3.8. Матрица неточностей, spread = 0.008, Ntrain = 2000.

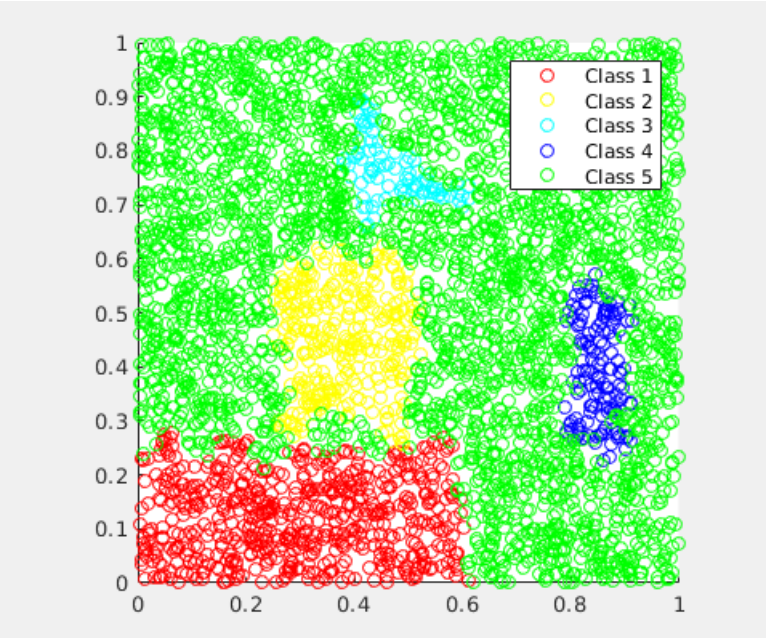

Рис. 3.9. Аппроксимация PNN, spread = 0.01, Ntrain = 500.

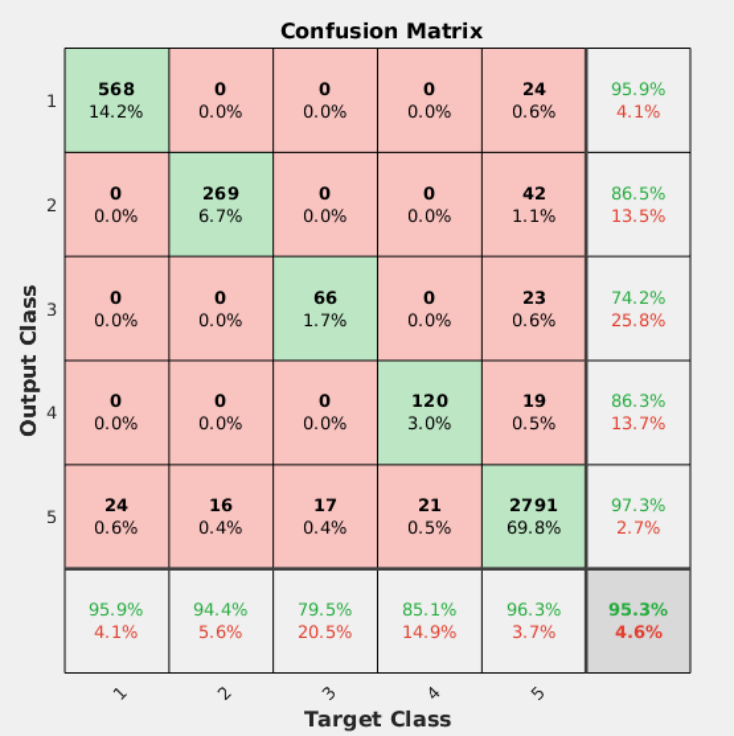

Рис. 3.10. Матрица неточностей, spread = 0.01, Ntrain = 500.

При уменьшении выборки значение *spread* пришлось увеличить;

Если сравнивать с классификацией НСПР, то выводы аналогичны выводам сравнения при классификации с 2 классами.

Построим поверхность ошибок. Для этой целей совсем немного была изменена программа из предыдущего задания.

*Программа **task3/errsurf***

```
function surface = errsurf(start_n, delta_n, end_n, start_s, delta_s, end_s)
    neurons = start_n:delta_n:end_n;
    spreads = start_s:delta_s:end_s;
    errs = zeros(length(neurons), length(spreads));
    for i = 1:length(neurons)
        for j = 1:length(spreads)
            [cross, ~] = task3(spreads(j), neurons(i), 0);
            errs(i, j) = cross;
        end
    end
    surface = errs;
    figure
    surf(neurons, spreads, errs,'FaceAlpha',0.8);
    colorbar
    xlabel('neurons');
    ylabel('spread');
    zlabel('crossentropy');
    hold on
    s = size(errs);
    z = zeros(s);
    surf(neurons, spreads, z, errs);
    xlim([start_n end_n]);
    ylim([start_s end_s]);
    view(127.5, 30)
```

Полученная в итоге поверхность показана на Рис. 3.11.

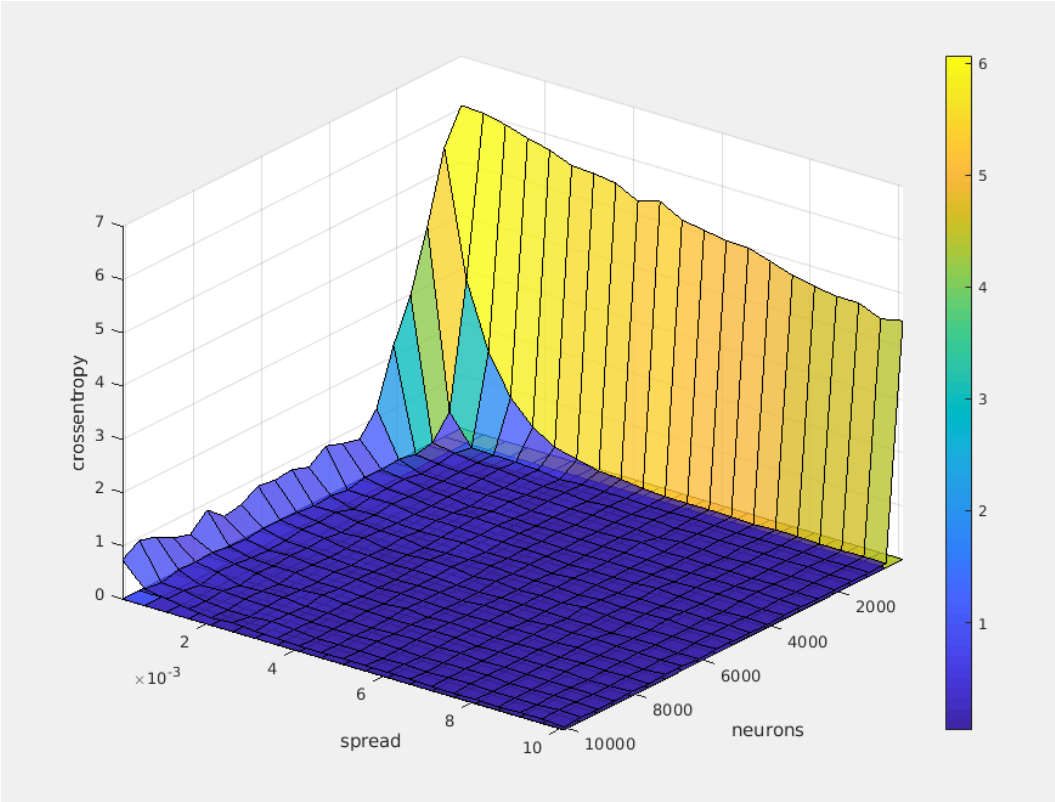

Рис. 3.11. Зависимость ошибки от числа нейноров и значения *spread*.