МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Кафедра вычислительной математики и программирования

Лабораторная работа №4
по спецкурсу «Нейроинформатика»

Сети с радиальными базисными элементами

Выполнил: Дюсекеев А.Е.
Группа: М8О-101М-21
Преподаватель: Леонов С.С.

Москва, 2022

Цель работы

Исследование свойств некоторых видов сетей с радиальными базисными элементами, алгоритмов обучения, а также применение сетей в задачах классификации и аппроксимации функции.

Основные этапы работы

1. Использовать вероятностную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.
2. Использовать сеть с радиальными базисными элементами (*RBF*) для классификации точек в случае, когда классы не являются линейно разделимыми.
3. Использовать обобщенно-регрессионную нейронную сеть для аппроксимации функции. Проверить работу сети с рыхлыми данными.

Оборудование

Параметры процессора:

Name	i9-12900K
Processor Base Frequency	3.20 GHz
Number of Cores	16

Оперативная память:

Всего	16.0 ГБ
Скорость	2133 МГц
Тип памяти	DDR4

Программное обеспечение

Matlab R2015b, 64-bit.

Сценарий выполнения работы

1. Для трёх линейно неразделимых классов из лабораторной работы №3 решить задачу классификации. Точки, принадлежащие одному классу, лежат на алгебраической линии. Построить вероятностную сеть, которая будет классифицировать точки заданной области.

   Обучающий набор $\{x_i, y_i\}, i = 1, \dots, N$, число классов $K = 3$. Сеть реализует отображение вида:

$$f(x_i, y_i) = \{(z_k)_{k=1}^{K} = (0, \dots, 1, \dots, 0) \big| z_{k=K^*} = 1 \text{ при } (x_i, y_i) \in K^*\}$$

$$t = 0 : 0.025 : 2\pi$$

$$x = f(t)$$

$$y = g(t)$$

   Эллипс: $a = 0.2, b = 0.2, \alpha = 0.2, x_0 = 0.2, y_0 = 0$;

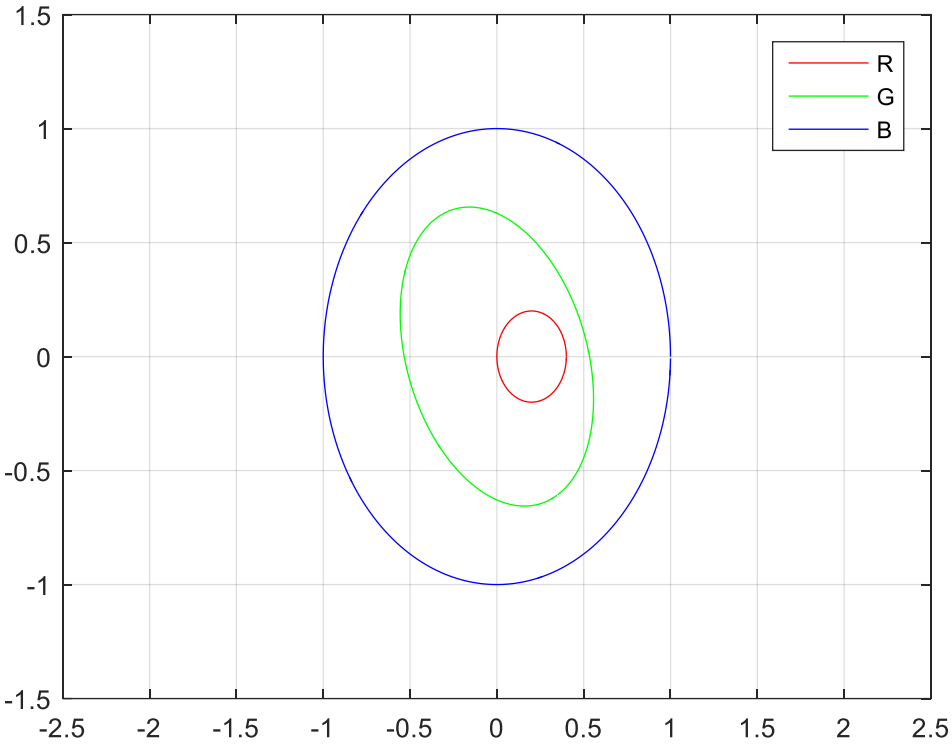
   Эллипс: $a = 0.7, b = 0.5, \alpha = -\frac{\pi}{3}, x_0 = 0, y_0 = 0$;

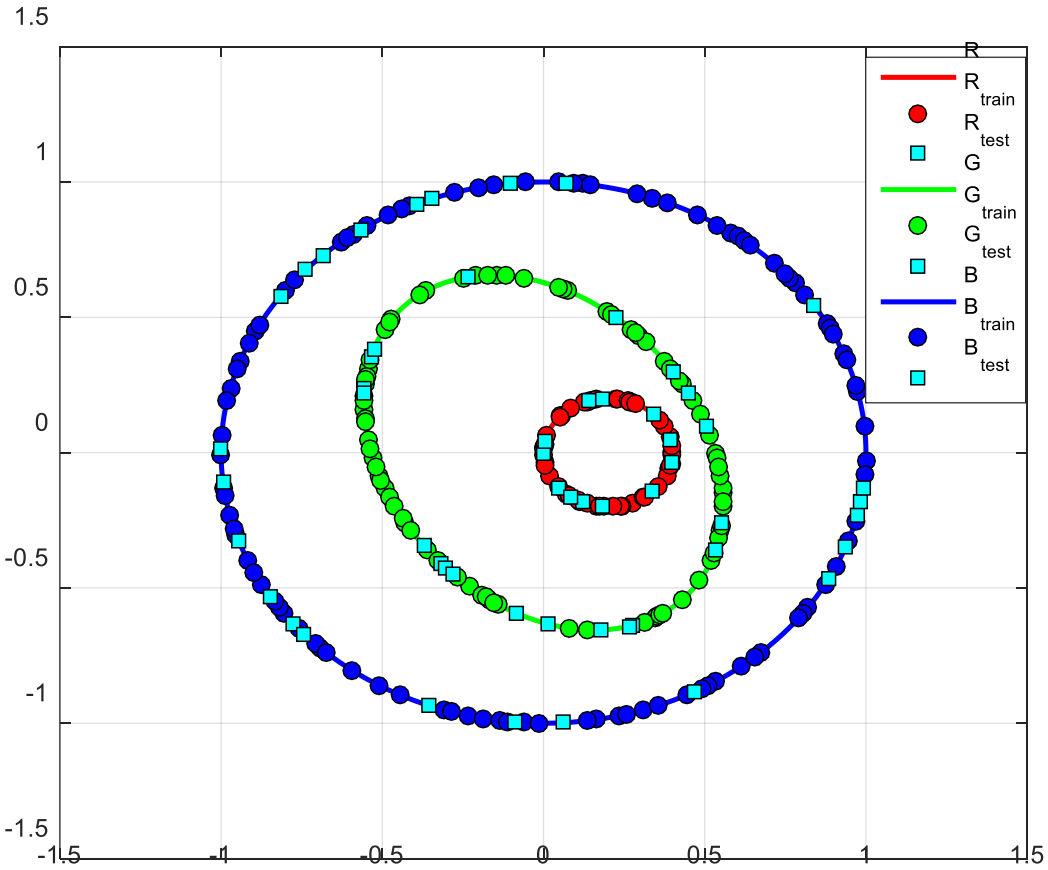
   Эллипс: $a = 1, b = 1, \alpha = 0, x_0 = 0, y_0 = 0$.

   1.1. В соответствии с вариантом задания для каждой линии сгенерировать множество точек. Далее для первого класса выбрать из исходного множества случайным образом 60 точек. Для второго и третьего классов 100 и 120 точек соответственно.

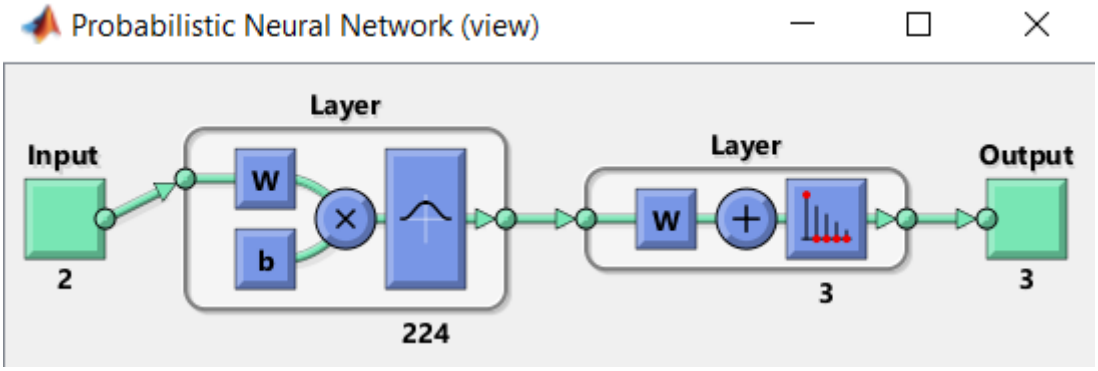
   1.2. Множество точек, принадлежащее каждому классу, разделить на обучающее и тестовое подмножества с помощью функции *dividerand* в отношении 80%-20%.

   1.3. Способом, описанным в Л.р. №3, отобразить множества точек для каждого класса, а также соответствующие обучающие и тестовые подмножества.


R
G
B
1.5
1
0.5
0
-0.5
-1
-1.5
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5


R
R train
R test
G
G train
G test
B
B train
B test
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5

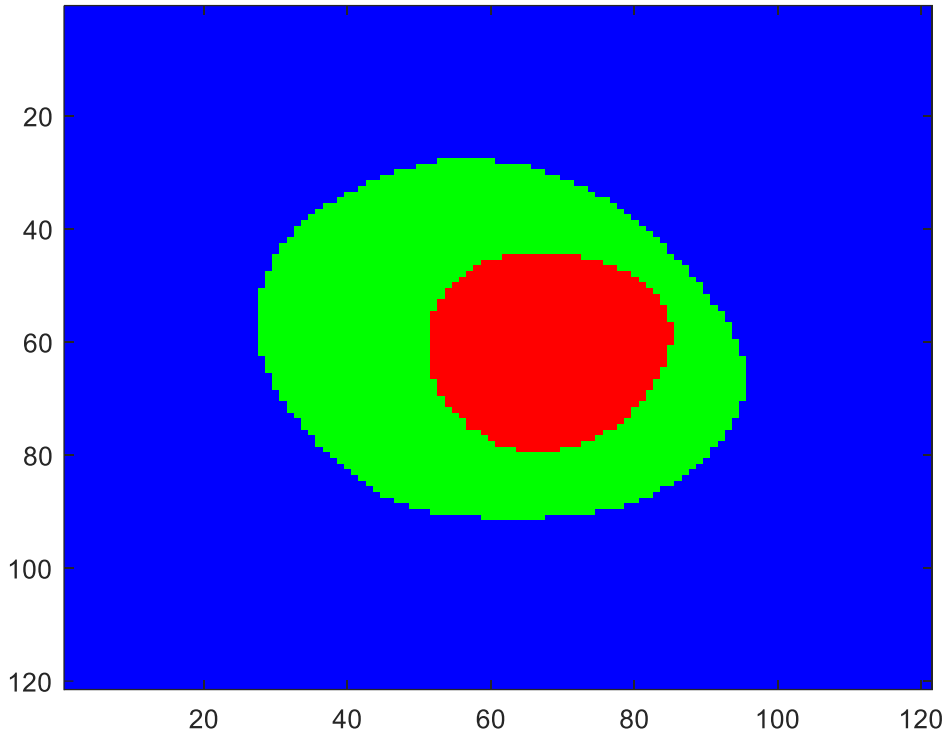
1.4. Соответствующие подмножества точек объединить в обучающее и тестовое подмножества обучающей выборки.
1.5. Эталонное распределение точек обучающей выборки по классам преобразовать к индексам (*ind2vec*).
1.6. Константу *SPREAD* задать равной 0.3. Создать сеть с помощью функции *newpnn*. Подать в сеть обучающее подмножество обучающей выборки.
1.7. Отразить структуру сети:



1.8. Проверить качество обучения: рассчитать выход сети для обучающего подмножества обучающей выборки. Преобразовать выходные значения с помощью функции (*vec2ind*). Занести в отчет количество правильно классифицированных точек.
1.9. Провести аналогичные расчеты для тестового подмножества.

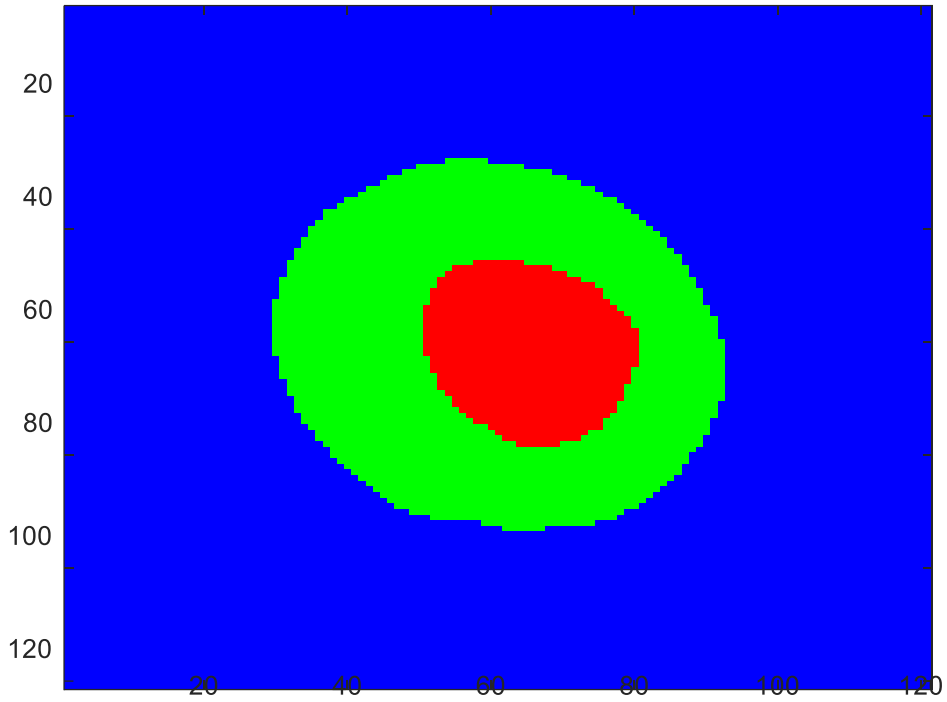
```
Обучающие: 212/224
Тестовые: 53/56
```

1.10. Произвести классификацию точек области $[-1.5, 1.5] \times [-1.5, 1.5]$. Закодировать принадлежности классам различными цветами и занести полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.

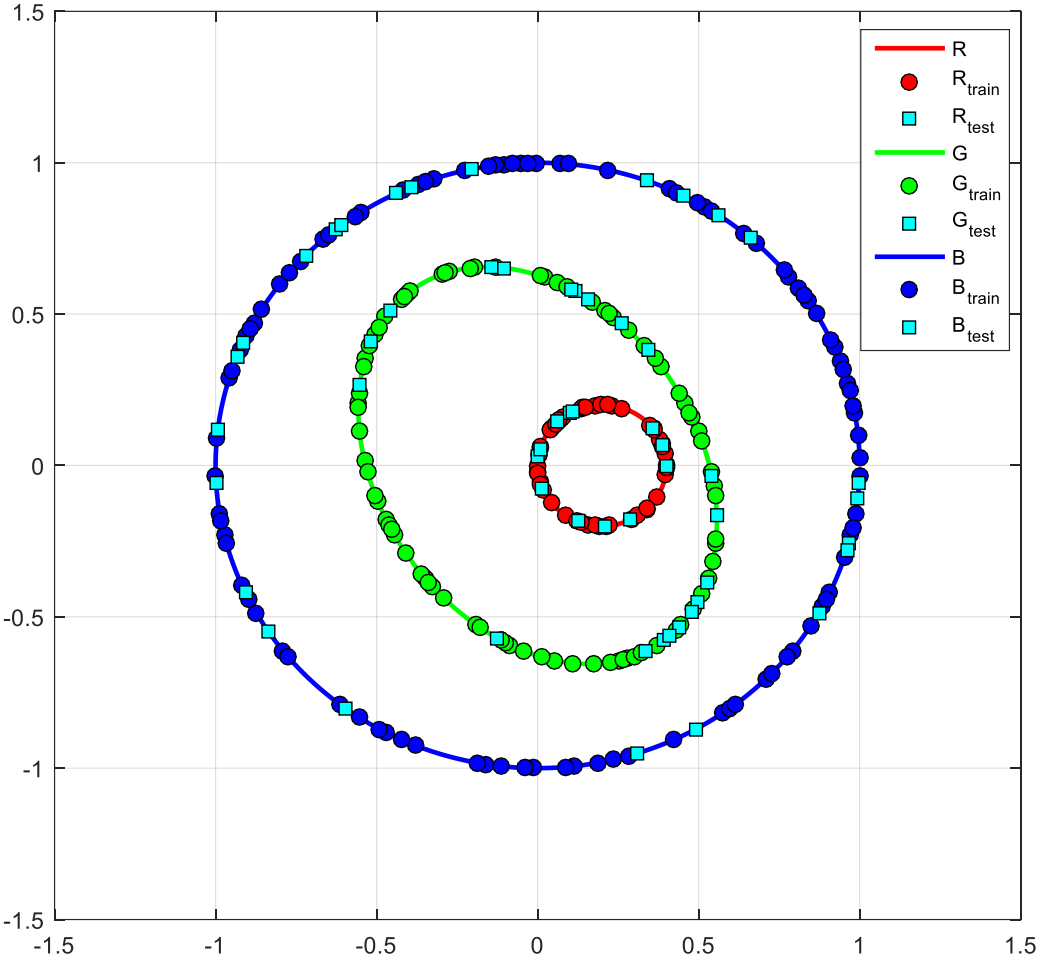


1.11. Константу *SPREAD* задать равной 0.1. Создать сеть с помощью функции *newpnn*.

1.12. Произвести классификацию точек области $[-1.5, 1.5] \times [-1.5, 1.5]$. Закодировать принадлежности классам различными цветами и занести полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.



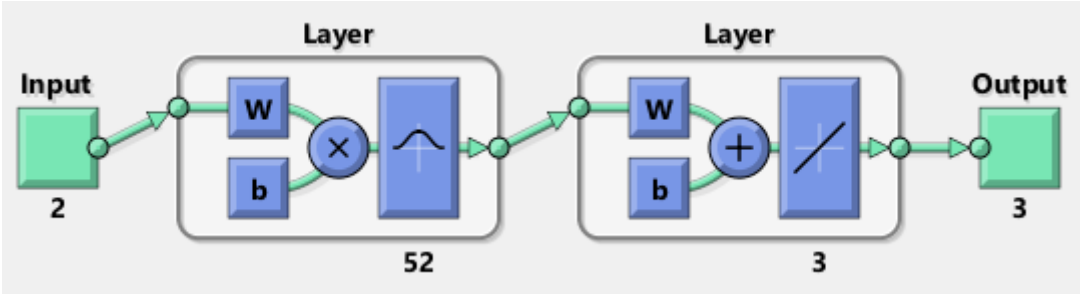
2. Для трех линейно неразделимых классов из лабораторной работы № 3 решить задачу классификации. Точки, принадлежащие одному классу, лежат на алгебраической линии. Построить сеть с радиальными базисными элементами, которая будет классифицировать точки заданной области.
   2.1. В соответствии с вариантом задания для каждой линии сгенерировать множество точек. Далее для первого класса выбрать из исходного множества случайным образом 60 точек. Для второго и третьего классов 100 и 120 точек соответственно.
   2.2. Множество точек, принадлежащее каждому классу, разделить на обучающее и тестовое подмножества с помощью функции *dividerand* в отношении 80%-20%.
   2.3. Способом, описанным в Л.р. №3, отобразить множества точек для каждого класса, а также соответствующие обучающие и тестовые подмножества.

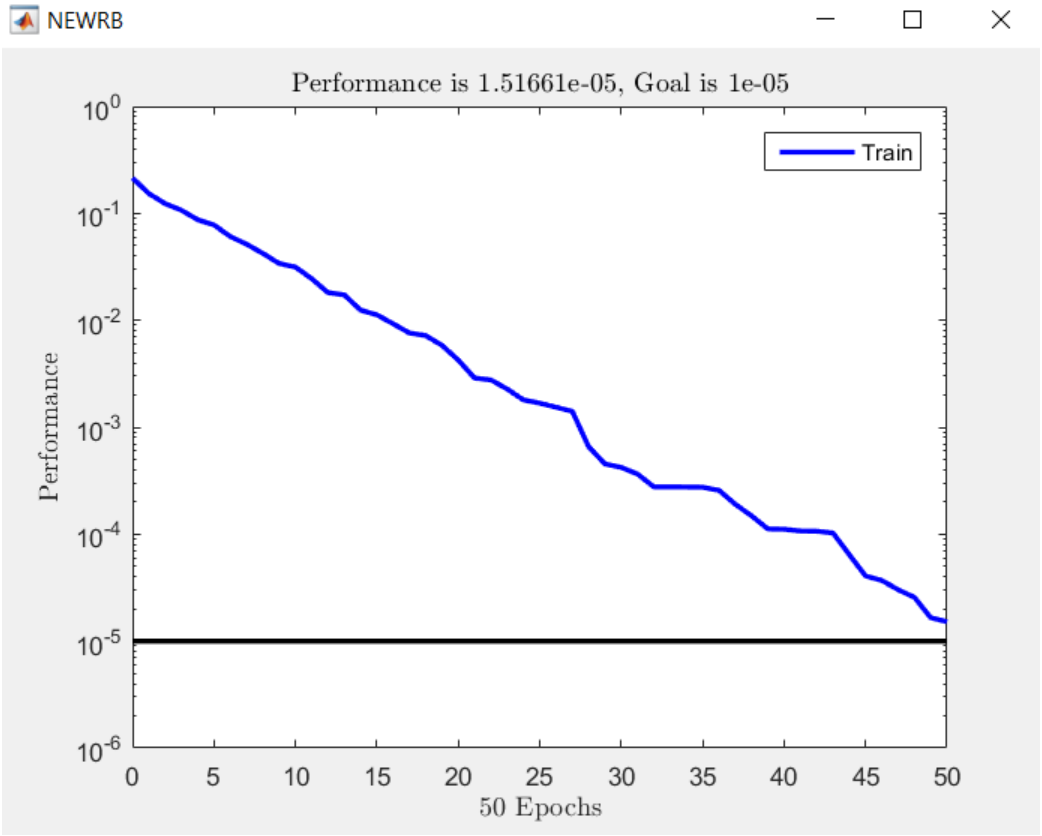


2.4. Соответствующие подмножества точек объединить в обучающее и тестовое подмножества обучающей выборки.

2.5. Создать сеть с помощью *newrb*, задав следующие параметры: предельное значение критерия обучения (*goal*) — 10^{-5}, *SPREAD* — 0.3, размер обучающей выборки — число элементов в обучающем подмножестве. В сеть подается обучающее подмножество обучающей выборки.

2.6. Занести в отчет окно *Training with newrb*. Отразить структуру сети. Указать число радиальных базисных нейронов.



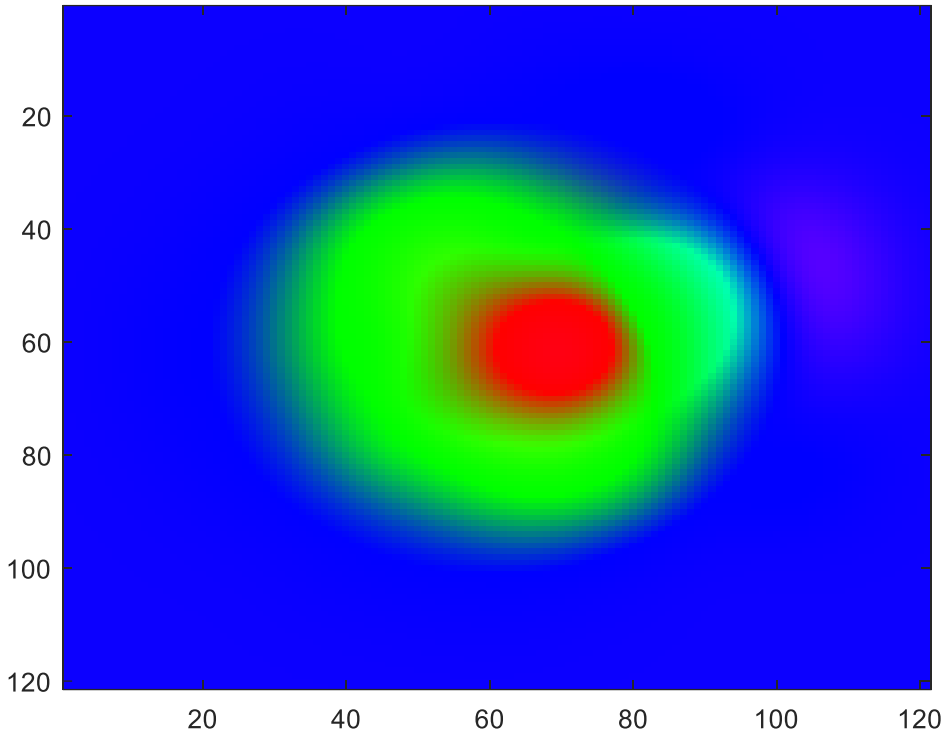


2.7. Проверить качество обучения: рассчитать выход сети для обучающего подмножества обучающей выборки. Занести в отчет количество правильно классифицированных точек.

2.8. Провести аналогичные расчеты для тестового подмножества.

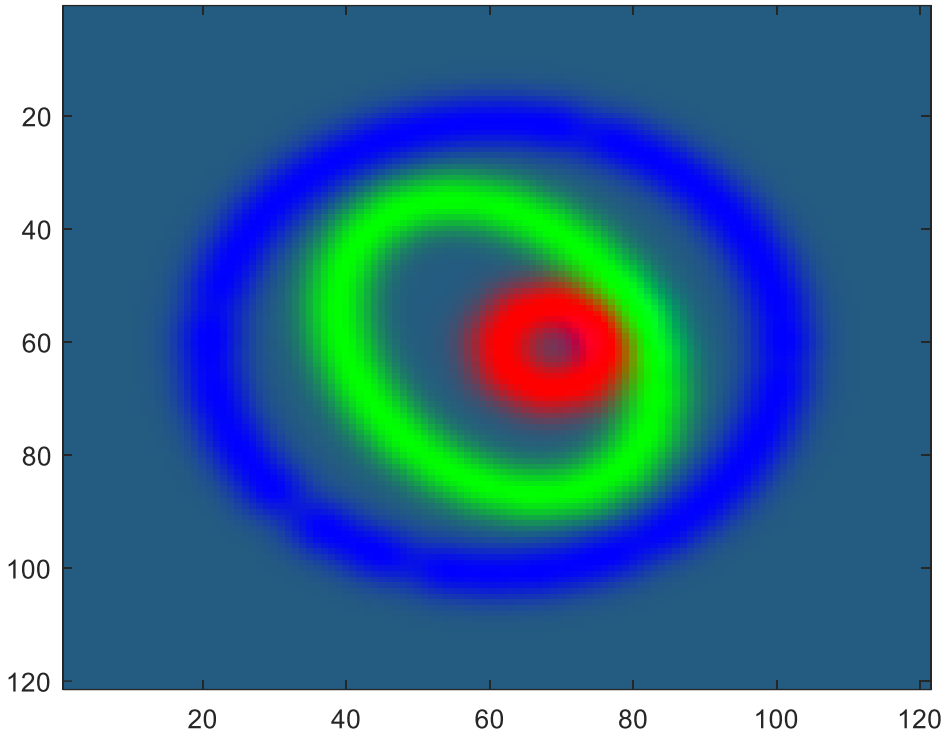
```
Обучающие: 224/224
Тестовые: 56/56
```

2.9. Произвести классификацию точек области $[-1.5, 1.5] \times [-1.5, 1.5]$. Закодировать принадлежности классам различными цветами и занести полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.



2.10. Константу *SPREAD* задать равной 0.1. Создать сеть с помощью функции *newrb*.

2.11. Произвести классификацию точек области $[-1.5, 1.5] \times [-1.5, 1.5]$. Закодировать принадлежности классам различными цветами и занести полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.



3. Задан обучающий набор $\{x(i), y(i)\}$. i)}. Построить и обучить двухслойную нейронную сеть прямого распространения, которая будет выполнять аппроксимацию функции вида

$$\hat{y}(i) = f\big(x(i)\big)$$

Функция и метод обучения определяются вариантом задания:

$$x = \sin(t^2 - 7), \qquad t \in [0{,}5], h = 0.025$$

3.1. Создать сеть с помощью функции *newgrnn*(*P1*, *T1*, *SPREAD*). Константу *SPREAD* задать равной h, где h — величина шага для заданной функции.

3.2. Произвести разделение обучающей выборки на обучающее и тестовое подмножества. Индексы обучающего подмножества использовать для создания сети.

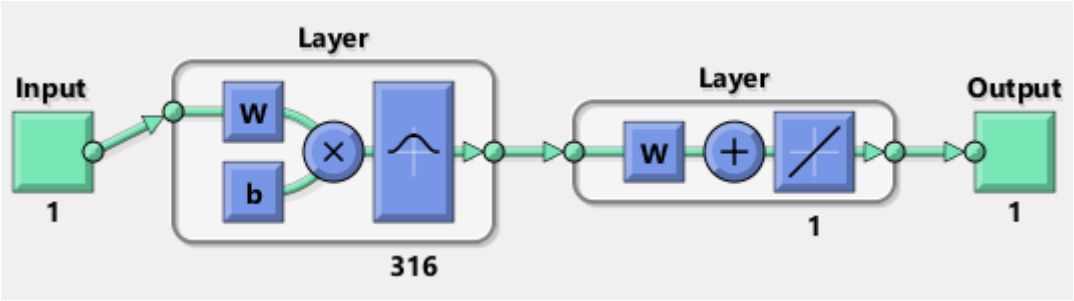
P1 = *P*(*trainInd*);

T1 = *T*(*trainInd*);

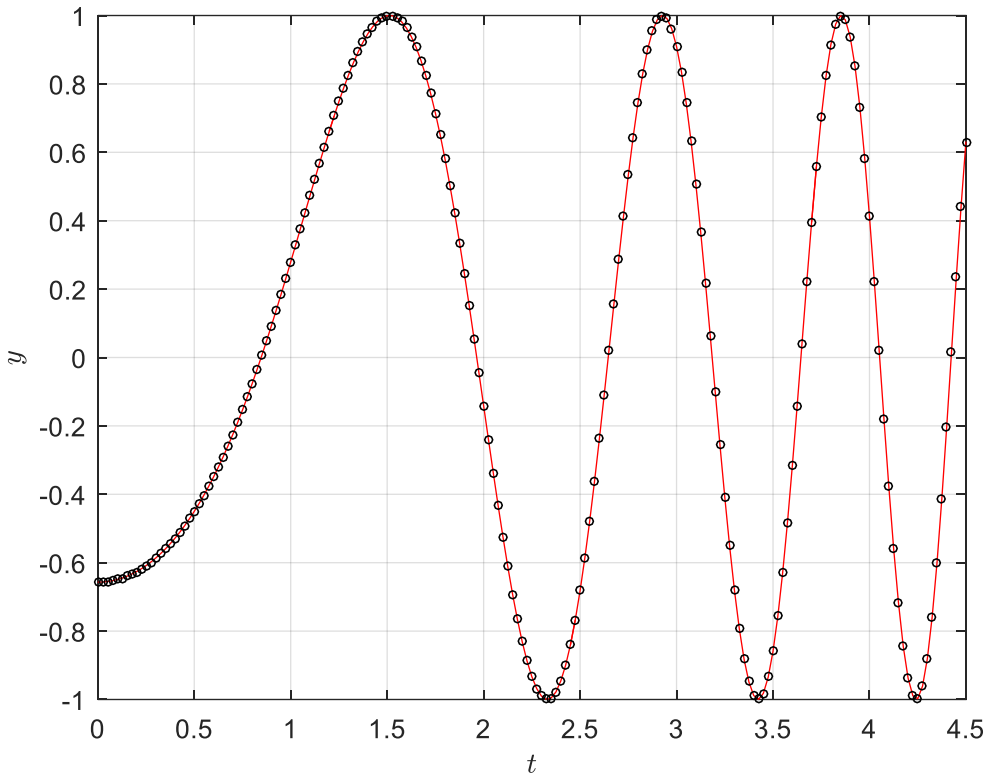
Выделить с конца временной последовательности 10% отсчетов на тестовое подмножество.

3.3. Если результаты неудовлетворительные, то изменить значение *SPREAD* и создать новую сеть.

3.4. Отразить структуру сети и проведенное обучение в отчете.

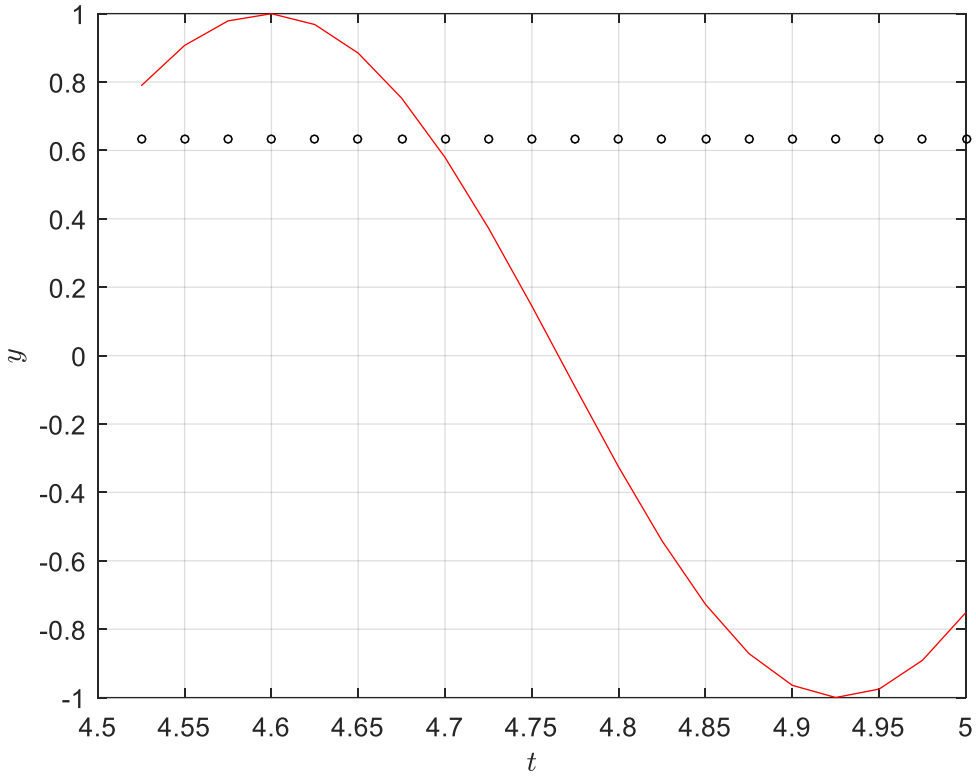


3.5. Рассчитать выход сети (*sim*) для обучающего подмножества. Сравнить выход сети с соответствующим эталонным подмножеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью. Отобразить на отдельном графике ошибку обучения. Графики занести в отчет.




```
R квадрат: 0.999980
MSE: 0.000010
RMSE: 0.003184
Относительная СКО: 0.159634%
MAE: 0.002299
min abs err: 0.000007
max abs err: 0.019362
MAPE: 0.929826
Доля с ошибкой менее 5%: 99.115044%
Доля с ошибкой от 5% до 10%: 0.442478%
Доля с ошибкой от 10% до 20%: 0.000000%
Доля с ошибкой от 20% до 30%: 0.000000%
Доля с ошибкой более 30%: 0.442478%
```

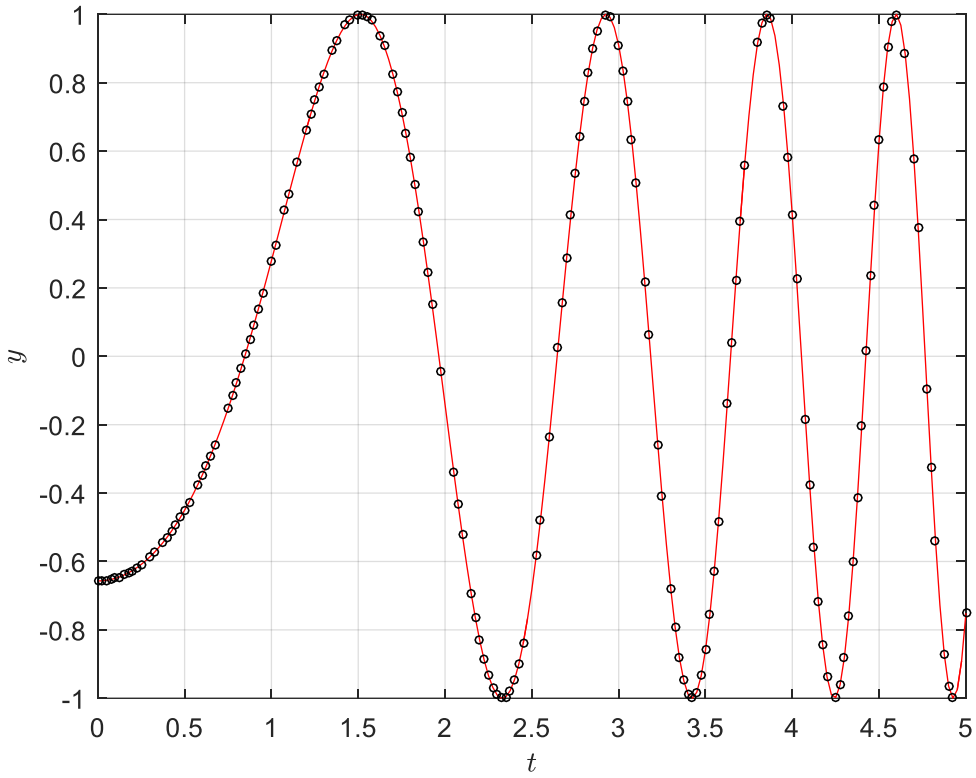
3.6. Получить апостериорную оценку качества работы сети: проделать аналогичные действия для тестового подмножества.



3.7. Сформировать обучающее множество с рыхлыми данными. Для этого произвести разделение обучающей выборки на обучающее и тестовое подмножества. с помощью функции (*dividerand*) в соотношении 80% и 20%.

3.8. Рассчитать выход сети (*sim*) для обучающего подмножества. Сравнить выход сети с соответствующим эталонным подмножеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью, а также ошибку обучения. Графики занести в отчет.

Обучающее



R квадрат: 0.999645

MSE: 0.000165

RMSE: 0.012830

Относительная СКО: 0.642945%

MAE: 0.007384

min abs err: 0.000003

max abs err: 0.051530

MAPE: 45.856802

Доля с ошибкой менее 5%: 86.567164%

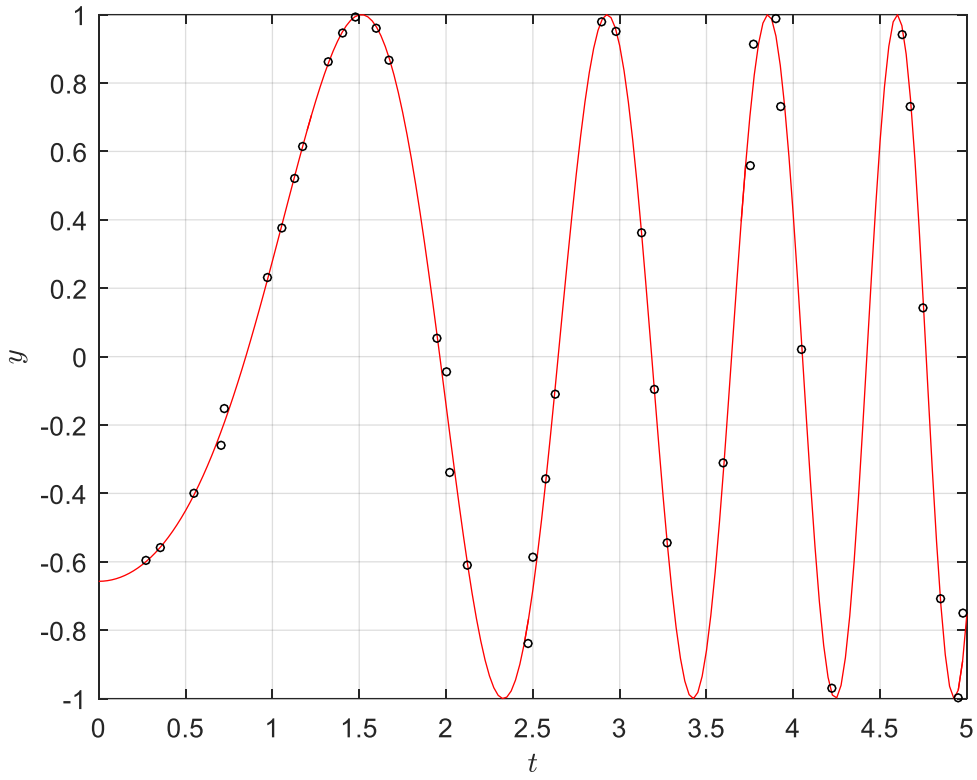
Доля с ошибкой от 5% до 10%: 5.970149%

Доля с ошибкой от 10% до 20%: 1.990050%

Доля с ошибкой от 20% до 30%: 0.497512%

Доля с ошибкой более 30%: 4.975124%

Тестовое




```
R квадрат: 0.996376

MSE: 0.001608

RMSE: 0.040101

Относительная СКО: 2.035185%

MAE: 0.021508

min abs err: 0.000028

max abs err: 0.104447

MAPE: 17.574318

Доля с ошибкой менее 5%: 78.000000%

Доля с ошибкой от 5% до 10%: 2.000000%

Доля с ошибкой от 10% до 20%: 6.000000%

Доля с ошибкой от 20% до 30%: 4.000000%

Доля с ошибкой более 30%: 10.000000%
```

Код программы

accuracy.m

```
function res = accuracy(y, yp)
% Высчитывание качественных характеристик обучения
    SSE = sum((y - yp) .^ 2);
```

```
    SSyy = sum((y - mean(y)) .^ 2);
    R_square = 1 - SSE/SSyy;

    MSE = mse(y - yp);

    RMSE = sqrt(MSE);

    СКО = RMSE / (max(y) - min(y)) * 100;

    MAE = mae(y - yp);

    MinAE = min(abs(y - yp));
    MaxAE =  max(abs(y - yp));

    MAPE = mean(abs((y - yp) ./ y)) * 100;

    errors = abs((y - yp) ./ y) * 100; % вектор относительных ошибок

    res = sprintf(['R квадрат: %f\n' ...
                   'MSE: %f\n' ...
                   'RMSE: %f\n' ...
                   'Относительная СКО: %f%%\n' ...
                   'MAE: %f\n'...
                   'min abs err: %f\n' ...
                   'max abs err: %f\n' ...
                   'MAPE: %f\n' ...
                   'Доля с ошибкой менее 5%%: %f%%\n' ...
                   'Доля с ошибкой от 5%% до 10%%: %f%%\n'...
                   'Доля с ошибкой от 10%% до 20%%: %f%%\n' ...
                   'Доля с ошибкой от 20%% до 30%%: %f%%\n' ...
                   'Доля с ошибкой более 30%%: %f%%\n'], ...
                  R_square, MSE, RMSE, СКО, MAE, MinAE, MaxAE, MAPE, ...
                  sum(errors < 5) / length(y) * 100, ...
                  sum(5 <= errors & errors < 10) / length(y) * 100, ...
                  sum(10 <= errors & errors < 20) / length(y) * 100, ...
                  sum(20 <= errors & errors < 30) / length(y) * 100, ...
                  sum(errors >= 30) / length(y) * 100);

end
```

main.m

```
% ЛР4
% Вариант 10

set(0, 'DefaultTextInterpreter', 'latex');

%% построение множества точек

t = 0:0.025:2*pi;

t = 0:0.025:2*pi;

alpha = 0;
x0 = 0.2;
y0 = 0.;
R = [cos(alpha), -sin(alpha); sin(alpha),  cos(alpha)] * [0.2 * cos(t); 0.2 *
sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

alpha = -pi/3;
x0 = 0.;
y0 = 0.;
```

```
G = [cos(alpha), -sin(alpha); sin(alpha),  cos(alpha)] * [0.7 * cos(t); 0.5 *
sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

alpha = 0.;
x0 = 0.;
y0 = 0.;
B = [cos(alpha), -sin(alpha); sin(alpha),  cos(alpha)] * [1 * cos(t); 1 *
sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

% нужно оставить только те точки, которые принадлежат области
cond = -1.5 <= B & B <= 1.5;
cond = cond(1, :) & cond(2, :);
B = B(:, cond);

plot(R(1, :), R(2, :), 'r', ...
     G(1, :), G(2, :), 'g', ...
     B(1, :), B(2, :), 'b');
legend('R', 'G', 'B');
axis([-2.5 2.5 -1.5 1.5]);
grid on;

%% формирование обучающего множества и разделение множества на обучающее,
контрольное и тестовое

r = R(:, randperm(end, 60));
g = G(:, randperm(end, 100));
b = B(:, randperm(end, 120));

[r_train, r_val, r_test] = dividerand(r, 0.8, 0.0, 0.2);
[g_train, g_val, g_test] = dividerand(g, 0.8, 0.0, 0.2);
[b_train, b_val, b_test] = dividerand(b, 0.8, 0.0, 0.2);

n_train = length(r_train) + length(g_train) + length(b_train);
n_val = length(r_val) + length(g_val) + length(b_val);
n_test = length(r_test) + length(g_test) + length(b_test);

%% отображение

p = plot(R(1, :),       R(2, :),       '-r', ...
         r_train(1, :), r_train(2, :), 'or',  ...
         r_test(1, :),  r_test(2, :),  'rs', ...
         G(1, :),       G(2, :),       '-g', ...
         g_train(1, :), g_train(2, :), 'og',  ...
         g_test(1, :),  g_test(2, :),  'gs', ...
         B(1, :),       B(2, :),       '-b', ...
         b_train(1, :), b_train(2, :), 'ob',  ...
         b_test(1, :),   b_test(2, :), 'bs');

p(1).LineWidth = 2;

p(2).MarkerEdgeColor = 'k';
p(2).MarkerFaceColor = 'r';
p(2).MarkerSize = 7;

p(3).MarkerEdgeColor = 'k';
```

```
p(3).MarkerFaceColor = 'c';
p(3).MarkerSize = 7;

p(4).LineWidth = 2;

p(5).MarkerEdgeColor = 'k';
p(5).MarkerFaceColor = 'g';
p(5).MarkerSize = 7;

p(6).MarkerEdgeColor = 'k';
p(6).MarkerFaceColor = 'c';
p(6).MarkerSize = 7;

p(7).LineWidth = 2;

p(8).MarkerEdgeColor = 'k';
p(8).MarkerFaceColor = 'b';
p(8).MarkerSize = 7;

p(9).MarkerEdgeColor = 'k';
p(9).MarkerFaceColor = 'c';
p(9).MarkerSize = 7;
axis([-1.5 1.5 -1.5 1.5]);
legend('R', 'R_{train}', 'R_{test}', ...
       'G', 'G_{train}', 'G_{test}', ...
       'B', 'B_{train}', 'B_{test}');
grid on;

%% объединение в выборки с метками

X_train = [r_train g_train b_train];
y_train = [1 * ones(1, length(r_train)) 2 * ones(1, length(g_train)) 3 *
ones(1, length(b_train))];

X_test = [r_test g_test b_test];
y_test = [1 * ones(1, length(r_test)) 2 * ones(1, length(g_test)) 3 * ones(1,
length(b_test))];


%% создание сети + обучение

SPREAD = 0.1;

net = newpnn(X_train, ind2vec(y_train), SPREAD);

view(net);

%% проверка качества

n_right_train = sum(vec2ind(sim(net, X_train)) == y_train);
n_right_test =  sum(vec2ind(sim(net, X_test)) == y_test);

fprintf('Обучающие: %d/%d\nТестовые: %d/%d\n', ...
        n_right_train, n_train, ...
        n_right_test, n_test);
```

```
%% пытаемся в картинку

h = 0.025;

n = int32((1.5 + 1.5) / h) + 1;

x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.5 + (double(i)-1)*h; ...
                             1.5 - (double(j)-1)*h];
    end
end

image(permute(reshape(sim(net, x), [3 n n]), [2 3 1]));

%% newrb

net = newrb(X_train, ind2vec(y_train), 1e-5, 0.1);

%% проверка качества

n_right_train = full(sum(sum((sim(net, X_train) >= 0.5) == ind2vec(y_train),
1) == 3));
n_right_test =  full(sum(sum((sim(net, X_test) >= 0.5) == ind2vec(y_test), 1)
== 3));

fprintf('Обучающие: %d/%d\nТестовые: %d/%d\n', ...
        n_right_train, n_train, ...
        n_right_test, n_test);

%% Аппроксимация функции

f = @(t) sin(t.^2-7);
t = 0:0.025:5;

X = t;
y = f(t);

%% Оставляем с конца 10%

n_train = ceil(length(X) * 0.9);

X_train = X(1:n_train);
y_train = y(1:n_train);
X_test = X(n_train+1:end);
y_test = y(n_train+1:end);

net = newgrnn(X_train, y_train, 0.01);

%% Метрики и графики для обучающего подмножества

disp(accuracy(sim(net, X_train), y_train));
```

```
p = plot(X_train, y_train, ...
         X_train, sim(net, X_train), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Метрики и графики для тестового подмножества

disp(accuracy(sim(net, X_test), y_test));

p = plot(X_test, y_test, ...
         X_test, sim(net, X_test), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Делим в соотношении

[ind_train, x, ind_test] = dividerand(1:length(X), 0.8, 0.0, 0.2);

n_train = length(ind_train);
n_test = length(ind_test);

X_train = X(ind_train);
y_train = y(ind_train);
X_test = X(ind_test);
y_test = y(ind_test);

net = newgrnn(X_train, y_train, 0.01);

%% Метрики и графики для обучающего подмножества

disp(accuracy(sim(net, X_train), y_train));

p = plot(X, y, ...
         X_train, sim(net, X_train), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Метрики и графики для тестового подмножества

disp(accuracy(sim(net, X_test), y_test));

p = plot(X, y, ...
```

```
          X_test, sim(net, X_test), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;
```

Выводы

В лабораторной работе было проведено исследование свойств некоторых видов сетей с радиальными базисными элементами, алгоритмов обучения, а также применение сетей в задачах классификации и аппроксимации функции.