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

# Лабораторная работа №4
# по спецкурсу «Нейроинформатика»

# Сети с радиальными базисными элементами

Выполнил: Днепров И.С.
Группа: М8О-407Б, вариант 10
Преподаватели: Тюменцев Ю.В.

Москва, 2020

**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**

Процессор: 2,4 GHz Intel Core 2 Duo

Оперативная память: 8 ГБ 1067 MHz DDR3

**Программное обеспечение**

*Matlab R2020b, 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 $\{x_i, y_i\}, i = 1,\ldots, N,$ число классов $K = 3$. Сеть реализует отображение вида:

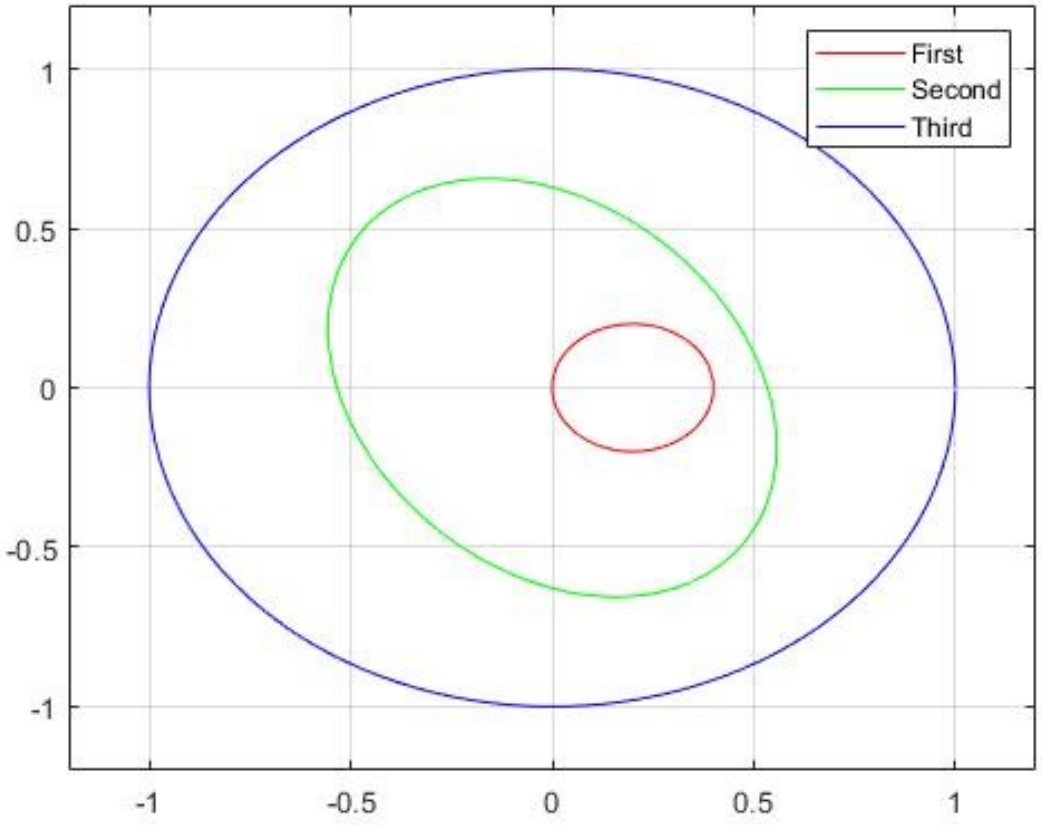
$$f(x_i, y_i) = \left\{ (z_k)_{k=1}^{K} = (0,\ldots, 1,\ldots, 0) \middle| z_{k=K^*} = 1 \; при \; (x_i, y_i) \in K^* \right\}$$

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.

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

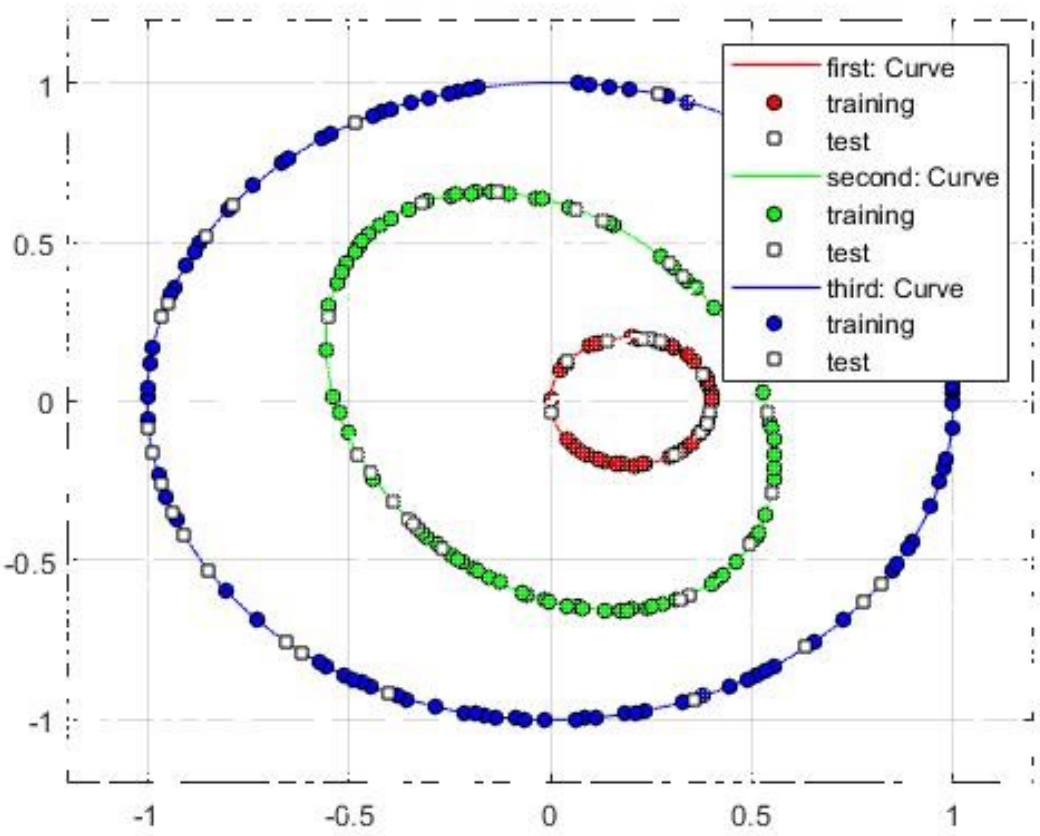
Эллипс: $a = 0.7, b = 0.5, \alpha = -\pi/3, x_0 = 0, y_0 = 0$

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




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.

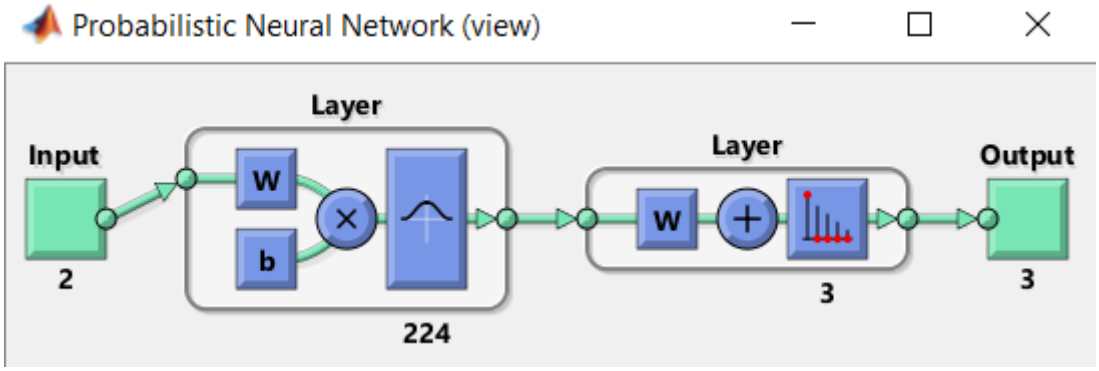



1.4. Соответствующие подмножества точек объединить в обучающее и тестовое подмножества обучающей выборки.

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:

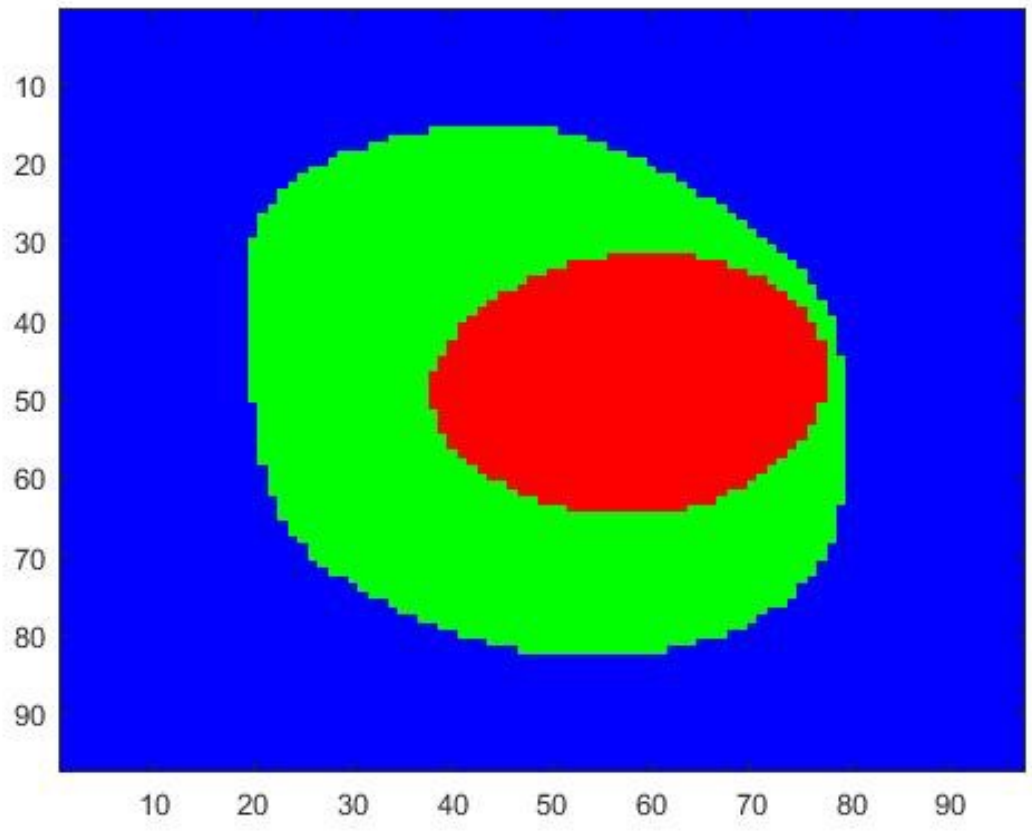



1.8. Проверить качество обучения: рассчитать выход сети для обучающего подмножества обучающей выборки. Преобразовать выходные значения с помощью функции (*vec2ind*). Занести в отчет количество правильно классифицированных точек.

1.9. Провести аналогичные расчеты для тестового подмножества.

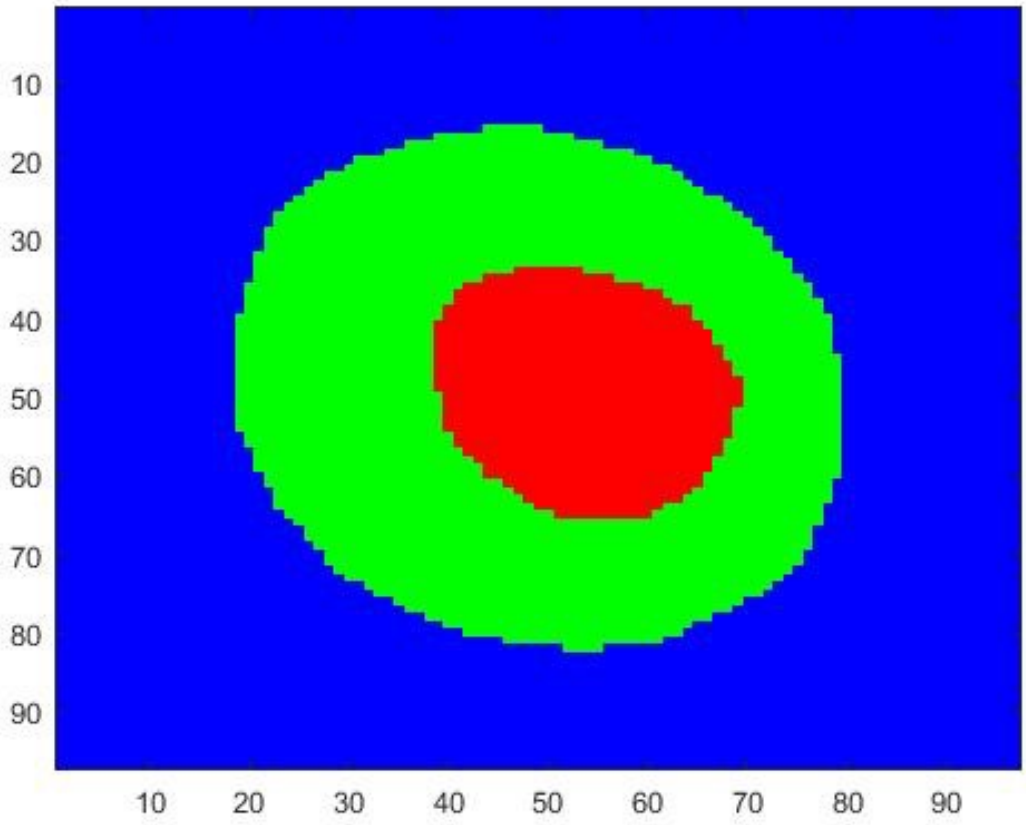
```
Обучающее множество: 207:224
Тестовое множество: 52:56
```

1.10. Произвести классификацию точек области $[-1.2, 1.2] \times [-1.2, 1.2]$. 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 $[-1.2, 1.2] \times [-1.2, 1.2]$. Закодировать принадлежности классам различными цветами и занести полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.




```
Обучающее множество: 224:224

Тестовое множество: 56:56
```

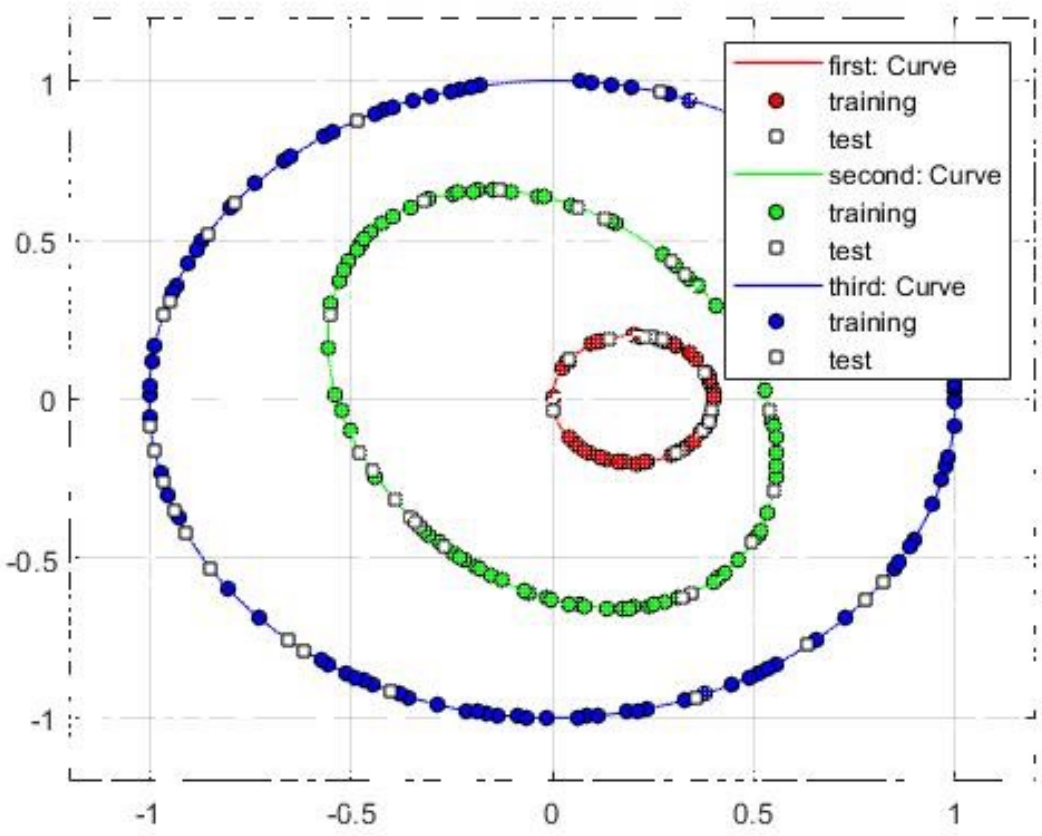
2. Для трех линейно неразделимых классов из лабораторной работы № 3 решить задачу классификации. Точки, принадлежащие одному классу, лежат на алгебраической линии. Построить сеть с радиальными базисными элементами, которая будет классифицировать точки заданной области.

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%.

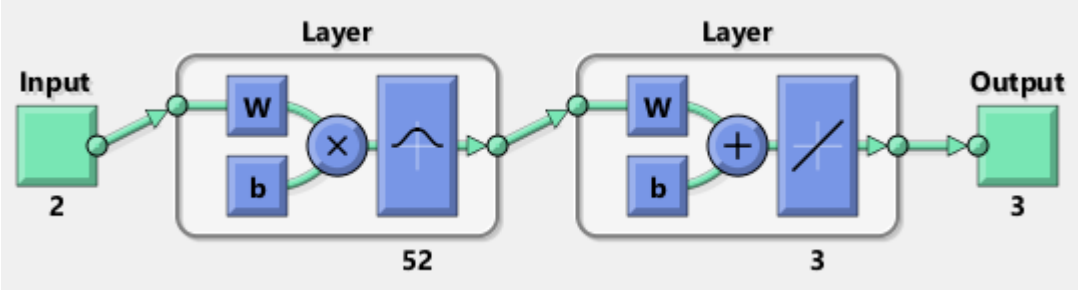
2.3. Способом, описанным в Л.р. №3, отобразить множества точек для каждого класса, а также соответствующие обучающие и тестовые подмножества.

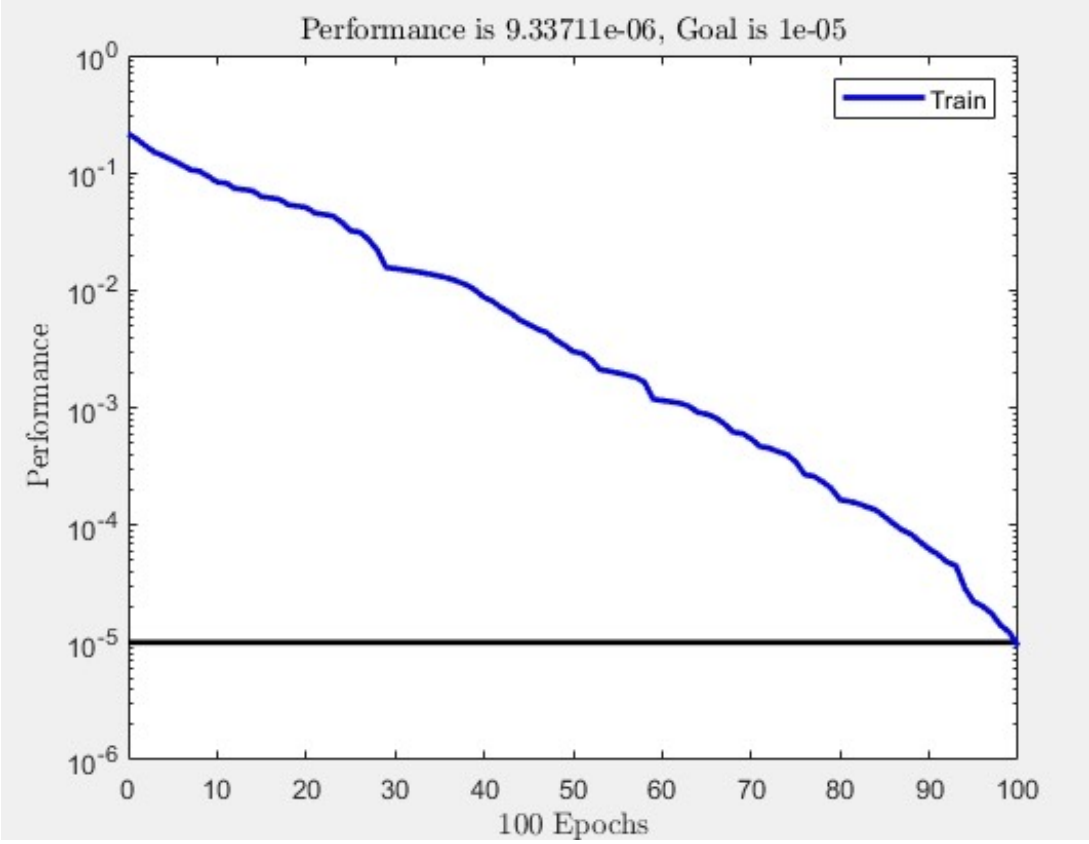



2.4. Соответствующие подмножества точек объединить в обучающее и тестовое подмножества обучающей выборки.

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.



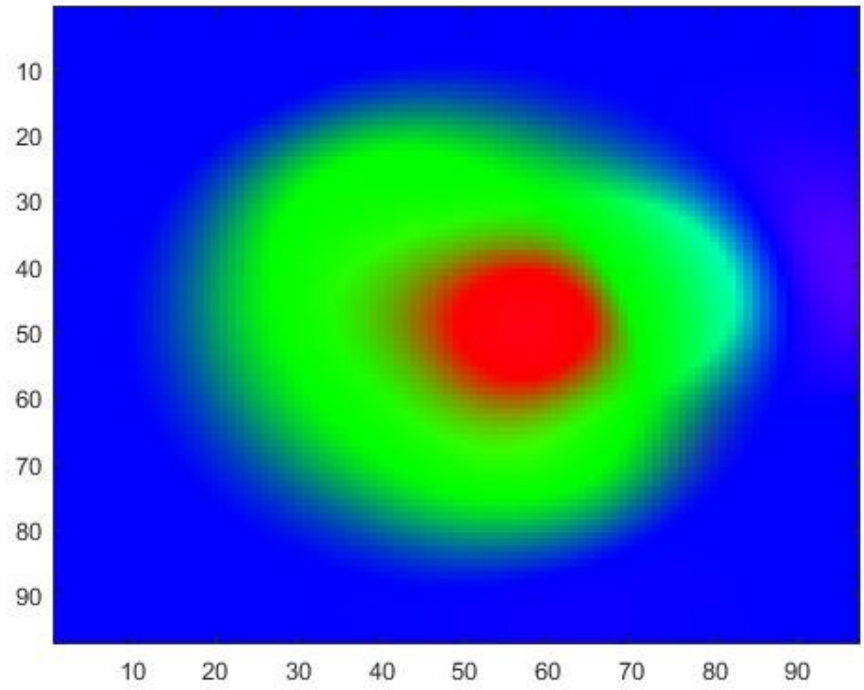




2.7. Проверить качество обучения: рассчитать выход сети для обучающего подмножества обучающей выборки. Занести в отчет количество правильно классифицированных точек.

2.8. Провести аналогичные расчеты для тестового подмножества.

```
Обучающее множество: 224:224
Тестовое множество: 56:56
```

2.9. Произвести классификацию точек области $[-1.2, 1.2] \times [-1.2, 1.2]$. Закодировать принадлежности классам различными цветами и занести



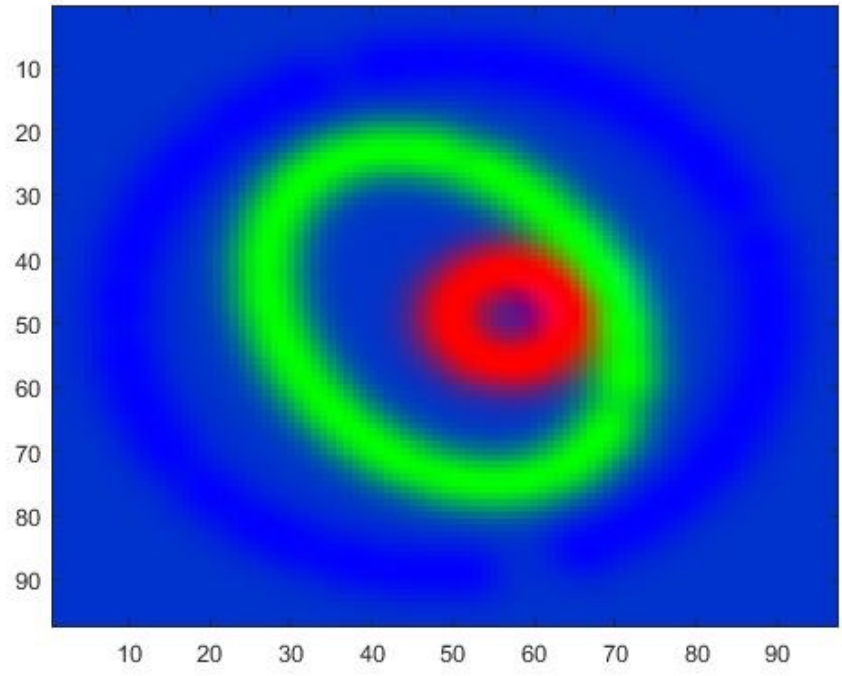

полученное изображение в отчёт. Для этого использовать методику, описанную в лабораторной работе №3.

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

```
Обучающее множество: 224:224

Тестовое множество: 56:56
```

2.11. Произвести классификацию точек области $[-1.2, 1.2] \times [-1.2, 1.2]$. 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

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

Функция и метод обучения определяются вариантом задания:

$x = \sin\left(t^2 - 7t\right), \quad t \in [0,5], h = 0.025$

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.

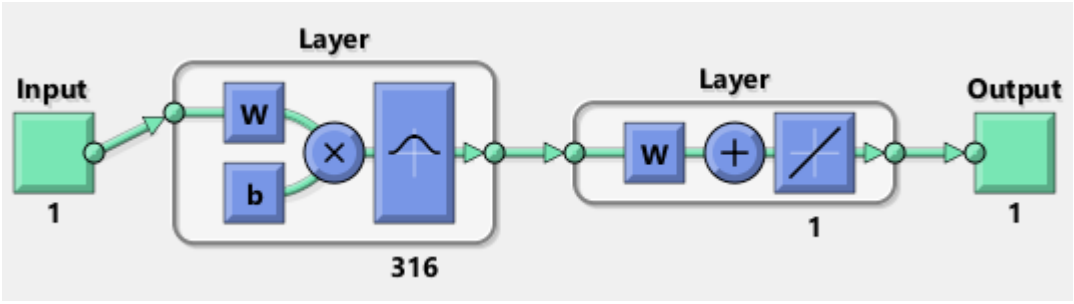
$$P1 = P(trainInd);$$

$$T1 = T(trainInd);$$

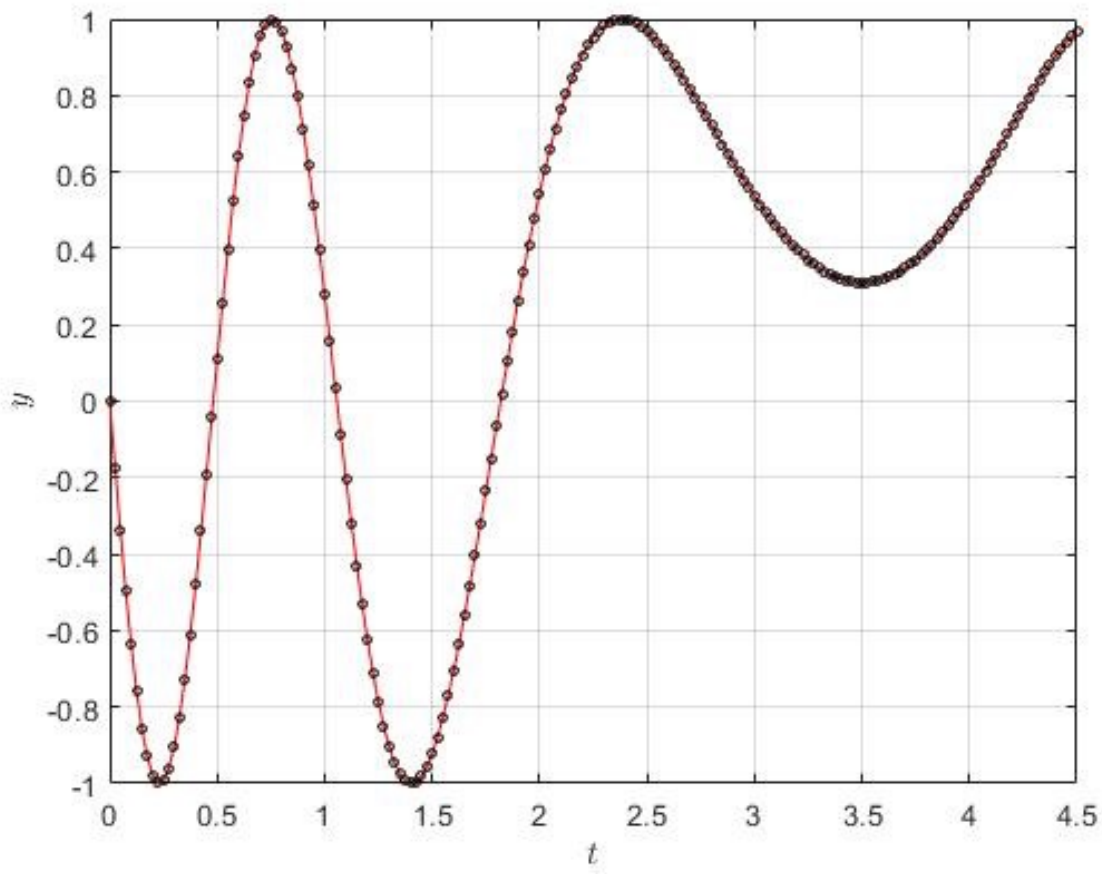
Выделить с конца временной последовательности 10% отсчетов на тестовое подмножество.

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

3.4. Отразить структуру сети и проведенное обучение в отчете.

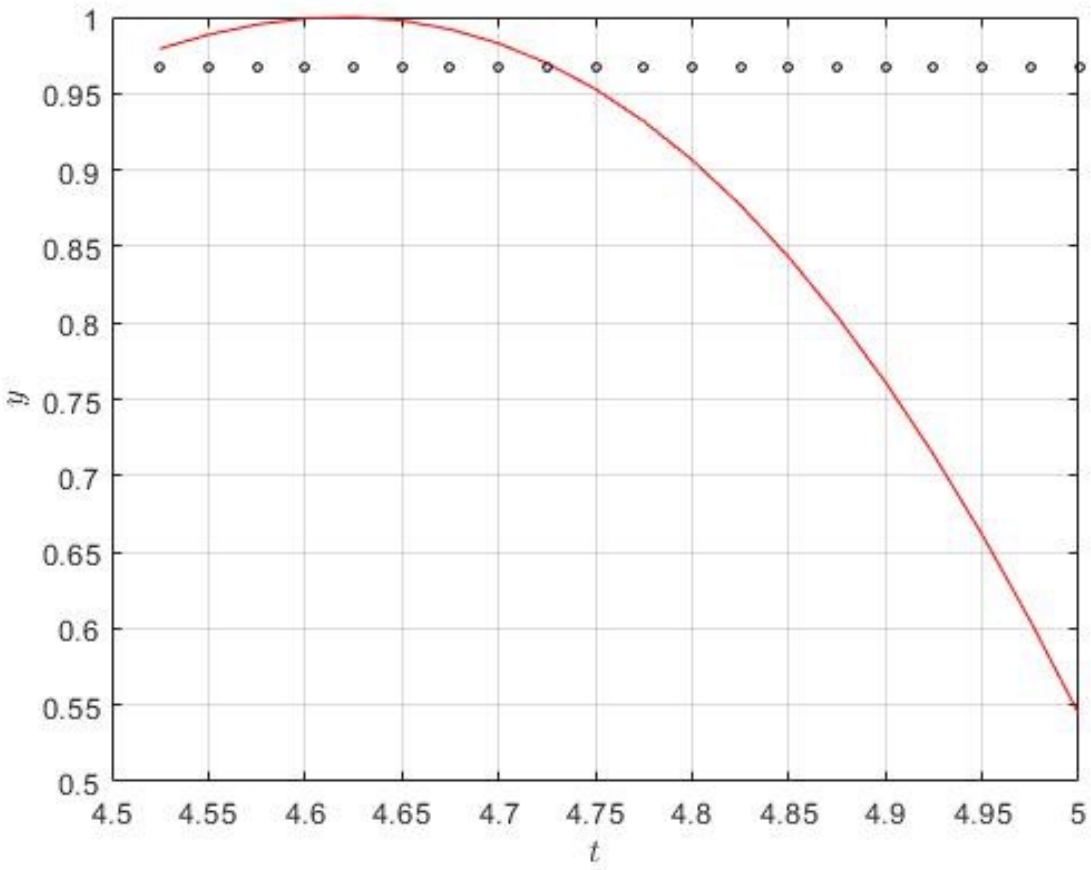



3.5. Рассчитать выход сети (*sim*) для обучающего подмножества. Сравнить выход сети с соответствующим эталонным подмножеством: рассчитать показатели качества обучения и заполнить таблицу 2. Отобразить на графике эталонные значения и предсказанные сетью. Отобразить на отдельном графике ошибку обучения.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




```
R^2: 1.000000
MSE: 0.000000
RMSE: 0.000200
Относительная СКО: 0.010020%
MAE: 0.000082
min abs err: 0.000000
max abs err: 0.002250
MAPE: 0.564164
Доля с ошибкой менее 5%: 99.447514%
Доля с ошибкой от 5% до 10%: 0.000000%
Доля с ошибкой от 10% до 20%: 0.000000%
Доля с ошибкой от 20% до 30%: 0.000000%
Доля с ошибкой более 30%: 0.552486%
```

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




R^2: -634059056116673.375000

MSE: 0.028666

RMSE: 0.169311

Относительная СКО: 548791933.847422%

MAE: 0.111950

min abs err: 0.002038

max abs err: 0.423787

MAPE: 11.567386

Доля с ошибкой менее 5%: 55.000000%

Доля с ошибкой от 5% до 10%: 10.000000%

Доля с ошибкой от 10% до 20%: 10.000000%

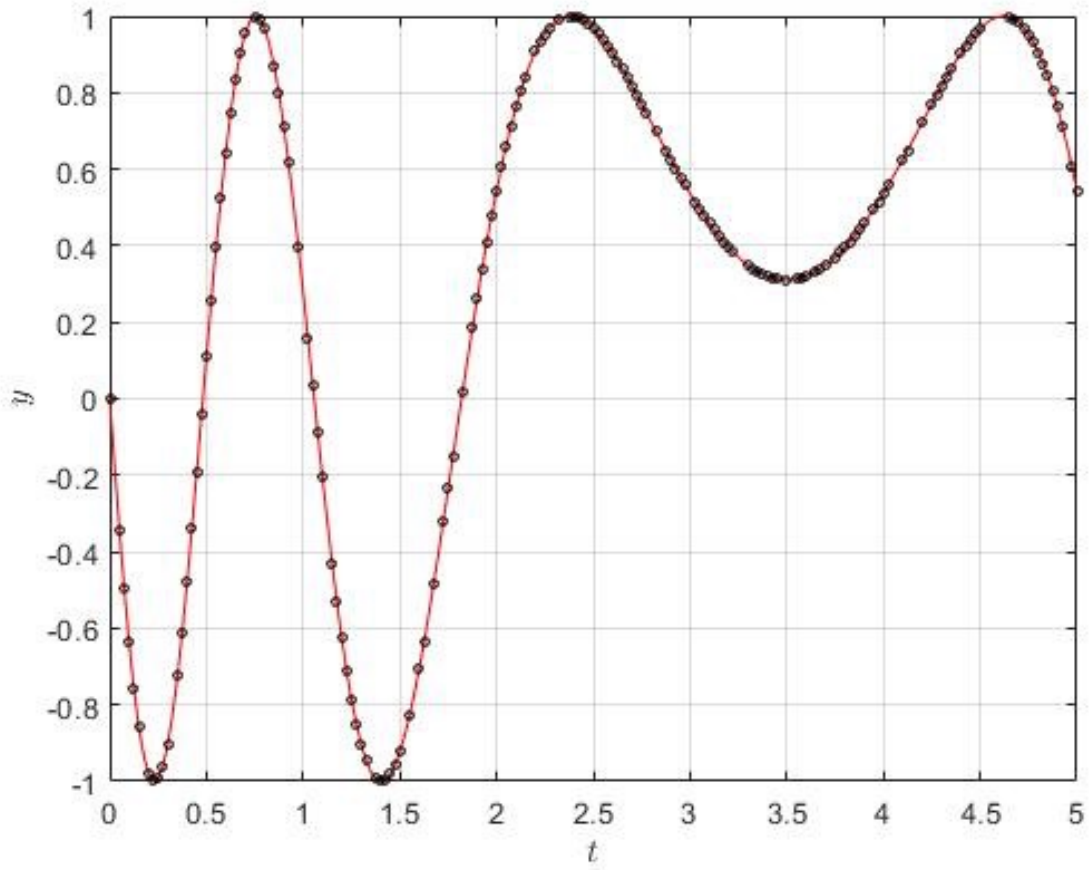
Доля с ошибкой от 20% до 30%: 10.000000%

Доля с ошибкой более 30%: 15.000000%

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 множество




R^2: 0.999999

MSE: 0.000000

RMSE: 0.000455

Относительная СКО: 0.022743%

MAE: 0.000236

min abs err: 0.000000

max abs err: 0.002250

MAPE: 0.683478

Доля с ошибкой менее 5%: 99.378882%

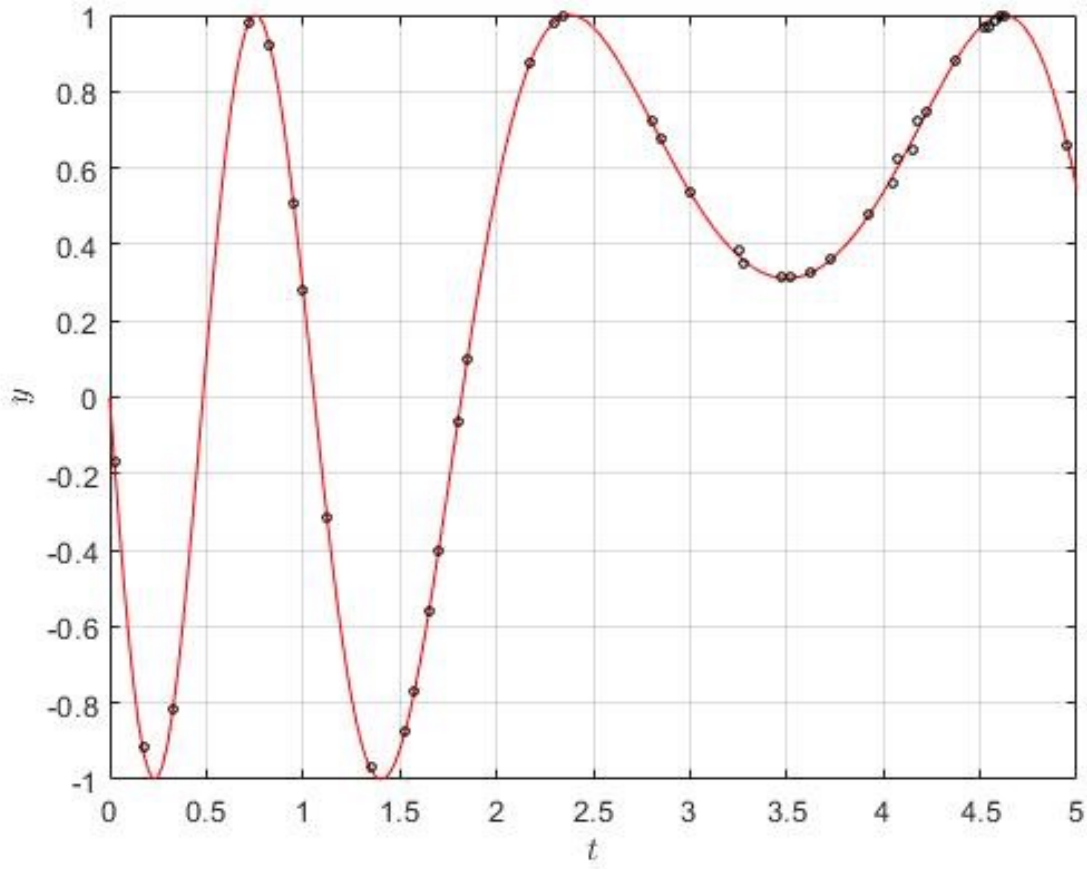
Доля с ошибкой от 5% до 10%: 0.000000%

Доля с ошибкой от 10% до 20%: 0.000000%

Доля с ошибкой от 20% до 30%: 0.000000%

Доля с ошибкой более 30%: 0.621118%

Тестовое множество




R^2: 0.999147

MSE: 0.000248

RMSE: 0.015763

Относительная СКО: 0.855423%

MAE: 0.010220

min abs err: 0.000059

max abs err: 0.060845

MAPE: 1.574866

Доля с ошибкой менее 5%: 97.500000%

Доля с ошибкой от 5% до 10%: 0.000000%

Доля с ошибкой от 10% до 20%: 2.500000%

Доля с ошибкой от 20% до 30%: 0.000000%

Доля с ошибкой более 30%: 0.000000%

**Код программы**

Lab4.m

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

% Этап 1
% Задаем три элипса
t = 0:0.025:2*pi;

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

a = 0.7;
b = 0.5;
alpha = -pi/3;
x0 = 0;
y0 = 0;
secondCurve = [cos(alpha), -sin(alpha); sin(alpha),  cos(alpha)] * [a * cos(t); b *
sin(t)] + [x0 * ones(1, length(t)); y0 * ones(1, length(t))];

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

% Показываем эти элипсы
plot(firstCurve(1, :), firstCurve(2, :), 'r', secondCurve(1, :), secondCurve(2, :),
'g', thirdCurve(1, :), thirdCurve(2, :), 'b');
legend('First', 'Second', 'Third');
axis([-1.2 1.2 -1.2 1.2]);
grid on;

%% Создаем множество точек и разделяем его
% Генерируем точки, принадлежашие элипсам
firstDots = firstCurve(:, randperm(end, 60));
secondDots = secondCurve(:, randperm(end, 100));
thirdDots = thirdCurve(:, randperm(end, 120));
```

```
% Делим точки
[firstTraining, firstControl, firstTest] = dividerand(firstDots, 0.8, 0.0, 0.2);
[secondTraining, secondControl, secondTest] = dividerand(secondDots, 0.8, 0.0, 0.2);
[thirdTraining, thirdControl, thirdTest] = dividerand(thirdDots, 0.8, 0.0, 0.2);

trainingSetSize = length(firstTraining) + length(secondTraining) +
length(thirdTraining);
n_val = length(firstControl) + length(secondControl) + length(thirdControl);
testSetSize = length(firstTest) + length(secondTest) + length(thirdTest);

% Демонстрация

p = plot(firstCurve(1, :), firstCurve(2, :), '-r', firstTraining(1, :), firstTraining(2,
:), 'or',  firstControl(1, :),   firstControl(2, :),   'rV', firstTest(1, :),
firstTest(2, :),  'rs', ...
         secondCurve(1, :), secondCurve(2, :), '-g', secondTraining(1, :),
secondTraining(2, :), 'og',  secondControl(1, :),   secondControl(2, :),   'gV',
secondTest(1, :),  secondTest(2, :),  'gs', ...
         thirdCurve(1, :), thirdCurve(2, :), '-b', thirdTraining(1, :),
thirdTraining(2, :), 'ob',  thirdControl(1, :),   thirdControl(2, :),   'bV',
thirdTest(1, :),   thirdTest(2, :), 'bs');

mSize = 5;
lWidth = 1;
edgeColor = 'black';
faceColor = 'white';

p(1).LineWidth = lWidth;
p(2).MarkerEdgeColor = edgeColor;
p(2).MarkerFaceColor = 'r';
p(2).MarkerSize = mSize;
p(3).MarkerEdgeColor = edgeColor;
p(3).MarkerFaceColor = faceColor;
p(3).MarkerSize = mSize;

p(4).LineWidth = lWidth;
p(5).MarkerEdgeColor = edgeColor;
p(5).MarkerFaceColor = 'g';
p(5).MarkerSize = mSize;
p(6).MarkerEdgeColor = edgeColor;
p(6).MarkerFaceColor = faceColor;
p(6).MarkerSize = mSize;

p(7).LineWidth = lWidth;
p(8).MarkerEdgeColor = edgeColor;
p(8).MarkerFaceColor = 'b';
p(8).MarkerSize = mSize;
p(9).MarkerEdgeColor = edgeColor;
p(9).MarkerFaceColor = faceColor;
p(9).MarkerSize = mSize;

axis([-1.2 1.2 -1.2 1.2]);
legend('first: Curve', 'training', 'test', ...
      'second: Curve', 'training', 'test', ...
      'third: Curve', 'training', 'test');
```

```
grid on;

%% Этапы 2 и 3
% Готовим данные для обучения
xTrainig = [firstTraining secondTraining thirdTraining];
yTrainig = [1 * ones(1, length(firstTraining)) 2 * ones(1, length(secondTraining)) 3
* ones(1, length(thirdTraining))];

xTest = [firstTest secondTest thirdTest];
yTest = [1 * ones(1, length(firstTest)) 2 * ones(1, length(secondTest)) 3 * ones(1,
length(thirdTest))];

% Создаем и обучаем сеть
SPREAD = 0.3;
network = newpnn(xTrainig, ind2vec(yTrainig), SPREAD);

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

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

%% Создаем и обучаем сеть
SPREAD = 0.1;
network = newpnn(xTrainig, ind2vec(yTrainig), SPREAD);

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
```

```
end

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

%% Создаем и обучаем сеть
SPREAD = 0.3;
network = newrb(xTrainig, ind2vec(yTrainig), 1e-5, SPREAD);

% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

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

%% Создаем и обучаем сеть
SPREAD = 0.1;
network = newrb(xTrainig, ind2vec(yTrainig), 1e-5, SPREAD);

%% Смотрим насколько хорошо сеть обучилась
correctTrainig = sum(vec2ind(sim(network, xTrainig)) == yTrainig);
correctTest = sum(vec2ind(sim(network, xTest)) == yTest);
fprintf('Обучающее множество: %d:%d\nТестовое множество: %d:%d\n', correctTrainig,
trainingSetSize, correctTest, testSetSize);

% Разделение по классам на картине
h = 0.025;
n = int32((1.2 + 1.2) / h) + 1;
x = zeros(2, n * n);

for i = 1:n
    for j = 1:n
        x(:, (i-1)*n + j) = [-1.2 + (double(i)-1)*h; 1.2 - (double(j)-1)*h];
    end
end

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

%% Этап 2 и 3
% Аппроксимация функции
f = @(t) sin(t .^ 2 - 7 * t);
t = 0:0.025:5;
```

```
X = t;
y = f(t);

% Оставляем с конца 10%
trainingSetSize = ceil(length(X) * 0.9);
xTrainig = X(1:trainingSetSize);
yTrainig = y(1:trainingSetSize);
xTest = X(trainingSetSize+1:end);
yTest = y(trainingSetSize+1:end);

% Создаем и обучаем нейросеть
network = newgrnn(xTrainig, yTrainig, 0.01);

%% Результаты обучения на обучающем подмножестве
disp(accuracy(sim(network, xTrainig), yTrainig));
p = plot(xTrainig, yTrainig, xTrainig, sim(network, xTrainig), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Результаты обучения на тестовом подмножестве
disp(accuracy(sim(network, xTest), yTest));
p = plot(xTest, yTest, xTest, sim(network, xTest), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;

%% Делим подмножества в заданном соотношении
[tariningSplit, x, testSplit] = dividerand(1:length(X), 0.8, 0.0, 0.2);
trainingSetSize = length(tariningSplit);
testSetSize = length(testSplit);
xTrainig = X(tariningSplit);
yTrainig = y(tariningSplit);
xTest = X(testSplit);
yTest = y(testSplit);
network = newgrnn(xTrainig, yTrainig, 0.01);

% Результаты обучения на обучающем подмножестве
disp(accuracy(sim(network, xTrainig), yTrainig));
p = plot(X, y, xTrainig, sim(network, xTrainig), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;
```

```
%% Результаты обучения на тестовом подмножестве
disp(accuracy(sim(network, xTest), yTest));
p = plot(X, y, xTest, sim(network, xTest), 'o');
p(1).Color = [1 0 0];
p(2).MarkerSize = 3;
p(2).Color = [0 0 0];
xlabel('$t$');
ylabel('$y$');
grid on;
```

## dataForTable.m

```
function res =  dataForTable(y, yp)
    R2 = 1 - sum((y - yp) .^ 2)/sum((y - mean(y)) .^ 2);
    MSE = mse(y - yp);
    RMSE = sqrt(MSE);
    CKO = RMSE / (max(y) - min(y)) * 100;
    MAE = mae(y - yp);
    MinAbsErr = min(abs(y - yp));
    MaxAbsErr =  max(abs(y - yp));
    MAPE = mean(abs((y - yp) ./ y)) * 100;
    errors = abs((y - yp) ./ y) * 100;
    Under5PersentPortion = sum(errors < 5) / length(y) * 100;
    Under10PersentPortion = sum(5 <= errors & errors < 10) / length(y) * 100;
    Under20PersentPortion = sum(10 <= errors & errors < 20) / length(y) * 100;
    Under30PersentPortion = sum(20 <= errors & errors < 30) / length(y) * 100;
    Over30PersentPortion = sum(errors >= 30) / length(y) * 100;

    res = sprintf(['R^2: %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'], ...
                  R2, MSE, RMSE, CKO, MAE, MinAbsErr, MaxAbsErr, MAPE,
Under5PersentPortion, Under10PersentPortion, Under20PersentPortion,
Under30PersentPortion, Over30PersentPortion);

end
```

**Вывод**

В данной работе работе было проведено исследование свойств некоторых видов сетей с радиальными базисными элементами. Благодаря визуализации обученной сети и вычислению количества верно классифицированных точек я имел возможность наглядно сравнить работу этих сетей и алгоритмов обучения. Также я имел возможность посмотреть на то, как ведут себя сети с радиальными базисными элементами при аппроксимации функции.

RBF (локально-рецептивные) сети в отличие от MLP сетей для любого входного вектора влиять на выходной результат сети будет ограниченное число нейронов. Таким образом эти сети быстрее переобучаются (обучение происходит не полное, а частичное). Таким образом MLP лучше чем RBF показывает себя на обучающей выборке, а на тестовой все происходит наоборот.