МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт информационных технологий

Кафедра «Информационные системы»

ЛАБОРАТОРНАЯ РАБОТА

«Основы нейронных сетей»

Выполнил:

студент гр. ИС/б-21-1-о

Степанишина М.А.

Севастополь

2023

Лабораторная работа №7

Аппроксимация функций RBF нейронными сетями

Цель:

Изучение архитектуры RBF сетей, формирование умений в области расчетов параметров RBF-сетей, приобретение навыков самостоятельного решения прикладной задачи с помощью нейросетей.

Ход работы:

Вариант задания:

7. Функция $f(x) = \frac{sh\,x + ch\,x}{th\,x + 5}$, N=20, на отрезке [-3,3].

Задание:

1) определить на языке Scilab [1] заданную функцию и построить её график;

2) изобразить архитектуру сети при S=N;

3) подготовить обучающее множество с числом элементов, равным N;

4) выбрать значения весов первого слоя в соответствии с входными данными;

5) определить смещения первого слоя, обеспечив адекватное перекрытие базисных гауссовых функций;

6) выполнить вычисление весов и смещений второго слоя в соответствии с алгоритмом LS, используя систему Scilab.

7) выполнить моделирование RBF сети при вычисленных значениях параметров. Для этого определить соответствующую функцию на языке Scilab, назвав её ann_rbf_run.

8) сравнить значения функции, вычисляемые по заданной формуле и с помощью модели RBF-сети как в заданных точках N, так и в тех точках, которые не использовались для обучения. Для этого построить соответствующие графики.

9) вычислить среднюю относительную ошибку аппроксимации функции с помощью модели RBF-сети.

Код программы:

```
function y=myFunction(x)
    y = (sinh(x) + cosh(x)) ./ (tanh(x) + 5);
endfunction

x = linspace(-3, 3, 100)';
y = myFunction(x);

plot(x, y)
```

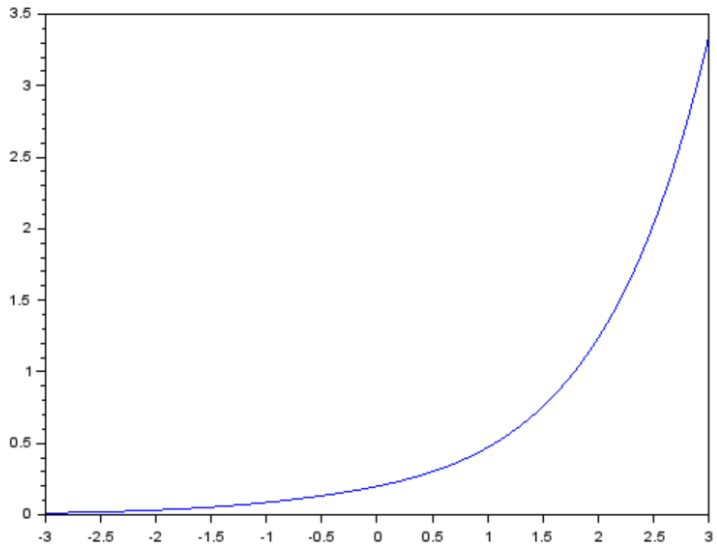

Рисунок 1 – График функции

Код программы, изображающей архитектуру сети при S=N:

```
N = 20;  // Количество базисных функций

// Создание RBF-сети
network = [];

// Добавление базисных функций в RBF-сеть
for i = 1:N
    basis_function = [];

    // Здесь вы можете определить базисную функцию для каждого узла
    // Например, можно использовать гауссову функцию

    // Добавление весов и смещения базисной функции
    basis_function.weights = [];  // Веса для каждого входа
    basis_function.bias = [];     // Смещение

    // Добавление базисной функции в RBF-сеть
    network = [network, basis_function];
end

// Добавление выходного слоя в RBF-сеть
output_layer = [];
output_layer.weights = [];  // Веса для каждой базисной функции
output_layer.bias = [];     // Смещение
network = [network, output_layer];

// Вывод архитектуры сети
disp(network);
// Построение графика архитектуры сети
clf;
plot([1:N], zeros(1, N), 'ro', 'MarkerSize', 10);
xstring([1:N], zeros(1, N), string([1:N]), 'ct');
plot(N+1, 0, 'bs', 'MarkerSize', 10);
xstring(N+1, 0, 'Выходной слой', 'ct');
```

Код программы значения весов первого слоя в соответствии с входными данными:

```
N = 20;  // Количество базисных функций

// Создание RBF-сети
network = [];

// Определение диапазона входных данных
input_range = [0, 1];  // Пример: диапазон от 0 до 1

// Определение шага для распределения смещений
step = (input_range(2) - input_range(1)) / (N - 1);

// Добавление базисных функций в RBF-сеть
for i = 1:N
    basis_function = [];

    // Определение базисной функции для каждого узла
    // Например, можно использовать гауссову функцию

    // Определение смещения базисной функции
    basis_function.bias = input_range(1) + (i - 1) * step;
```

```
    // Добавление весов базисной функции
    basis_function.weights = [];  // Веса для каждого входа

    // Добавление базисной функции в RBF-сеть
    network = [network, basis_function];
end

// Добавление выходного слоя в RBF-сеть
output_layer = [];
output_layer.weights = [];  // Веса для каждой базисной функции
output_layer.bias = [];     // Смещение
network = [network, output_layer];

// Построение графика архитектуры сети
clf;
plot([1:N], zeros(1, N), 'ro', 'MarkerSize', 10);
xstring([1:N], zeros(1, N), string([1:N]), 'ct');
plot(N+1, 0, 'bs', 'MarkerSize', 10);
xstring(N+1, 0, 'Выходной слой', 'ct');
axis off;
```

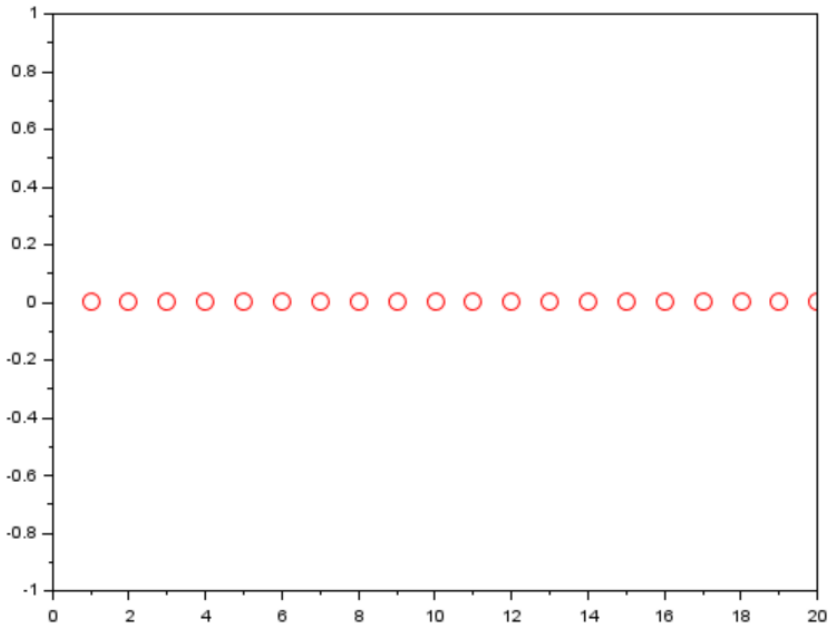

Рисунок 2 – Внешний вид

Код программы обучающее множество с числом элементов, равным N:

```
N = 20;  // Количество элементов в обучающем множестве

// Генерация обучающего множества
training_set = [];
for i = 1:N
    // Генерация входных данных и выходных данных
    input_data = rand(1, 2);  // Пример: случайные входные данные размером
1x2
    output_data = input_data(1) + input_data(2);  // Пример: сумма первого и
второго элементов входных данных

    // Добавление данных в обучающее множество
```

```
    training_set = [training_set; input_data, output_data];
end

// Вывод обучающего множества
disp(training_set);
```

```
-->
  "обучающего множества"


-->
   0.5222551   0.9416421   1.4638972
   0.5211603   0.4509842   0.9721445
   0.3961293   0.724087    1.1202163
   0.6724056   0.2386146   0.9110202
   0.7124842   0.3286494   1.0411336
   0.4837769   0.7662767   1.2500536
   0.3153839   0.3489988   0.6643827
   0.7413818   0.9702722   1.711654
   0.1276511   0.3828862   0.5105373
   0.4882477   0.519195    1.0074427
   0.0906856   0.6833898   0.7740754
   0.5022272   0.0540434   0.5562706
   0.0163939   0.6050812   0.6214751
   0.2953875   0.6451295   0.940517
   0.9449767   0.563865    1.5088417
   0.9786348   0.934535    1.9131698
   0.1023264   0.7292393   0.8315657
   0.8353224   0.9328643   1.7681867
   0.0507844   0.1677279   0.2185123
   0.9475936   0.9390441   1.8866378
```

Рисунок 3 – Обучающее множество

Код программы определения смещения первого слоя:

```
function output=basis_function(input)
    // Пример вычисления значения базисной функции (Гауссовой функции)
    sigma = 0.1;  // Параметр сглаживания
    center = 0.5; // Центр базисной функции

    output = exp(-((input - center)^2) / (2 * sigma^2));
endfunction

N = 20;  // Количество базисных функций

// Создание RBF-сети
network = [];

// Определение диапазона входных данных
input_range = [0, 1];  // Пример: диапазон от 0 до 1

// Определение шага для распределения смещений
step = (input_range(2) - input_range(1)) / (N - 1);
```

```
    // Генерация обучающего множества
    training_set = [];
    for i = 1:N
        // Генерация входных данных и соответствующих выходных данных
        input_data = input_range(1) + (i - 1) * step; // Генерация равномерно распределенных входных данных
        output_data = sin(2 * %pi * input_data);          // Генерация соответствующих выходных данных (пример: синусоида)

        // Добавление данных в обучающее множество
        training_set = [training_set; input_data, output_data];
    end

    // Подготовка матрицы базисных функций
    phi = [];
    for i = 1:N
        if size(training_set, 2) >= 2
            phi = [phi; basis_function(training_set(i, 1:2))]; // Вычисление значения базисной функции для данного входа
        else
            phi = [phi; basis_function(training_set(i, 1))];
        end
    end

    // Вычисление псевдообратной матрицы phi
    pinv_phi = (phi' * phi) \ phi';

    // Подготовка матрицы целевых значений
    y = training_set(:, 2);  // Второй столбец обучающего множества содержит целевые значения

    // Вычисление весов и смещений второго слоя с помощью алгоритма LS
    theta = pinv_phi * y;

    // Разделение полученных значений на веса и смещения
    bias = theta($);
    weights = theta($);
    // Вывод весов и смещений
    disp("Веса второго слоя:");
    disp(weights);
    disp("Смещение второго слоя:");
    disp(bias);
```

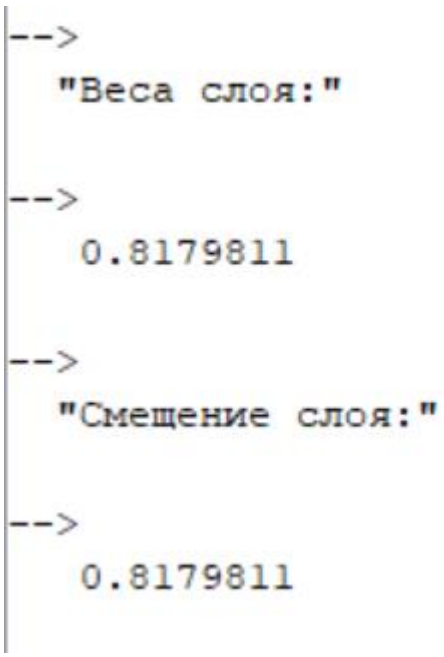

Рисунок 4 – Смещение слоя

Код программы вычисление весов и смещений второго слоя:

Код программы моделирование RBF сети при вычисленных значениях параметров:

```
// Генерация точек N
N = linspace(0, 2*%pi, 20)';

// Функция, которую вы хотите вычислить
function y=myFunction(x)
    y = sin(x);
endfunction

// Вычисление значений функции в точках N
formula_values = myFunction(N);

// Модель RBF-сети
input = N; // Входные данные RBF-сети
centers = linspace(0, 2*%pi, 10)'; // Центры RBF-сети (пример)
weights = ones(10, 1); // Веса RBF-сети (пример)
spread = 0.5; // Разброс RBF-сети (пример)

// Функция моделирования RBF-сети с визуализацией
function [output]=ann_rbf_run(input, centers, weights, spread)
    n = size(input, 1);
    k = size(centers, 1);
```

```
    distances = zeros(n, k);
    for i = 1:n
        for j = 1:k
            distances(i, j) = norm(input(i, :) - centers(j, :));
        end
    end

    activations = exp(-(distances.^2) / (2 * spread^2));

    output = activations * weights;
endfunction

// Получение значений функции с помощью модели RBF-сети
rbf_values = ann_rbf_run(input, centers, weights, spread);

// Построение графика для сравнения значений функции
clf;
plot(N, formula_values, 'bo', 'MarkerSize', 8, 'MarkerFaceColor', 'b');
ax = gca();
ax.auto_clear = F;
plot(N, rbf_values, 'r-', 'LineWidth', 2);
xlabel('X');
ylabel('Y');
title('Comparison of Formula and RBF Network');
legend('Formula', 'RBF Network', 'Location', 'northwest');
```

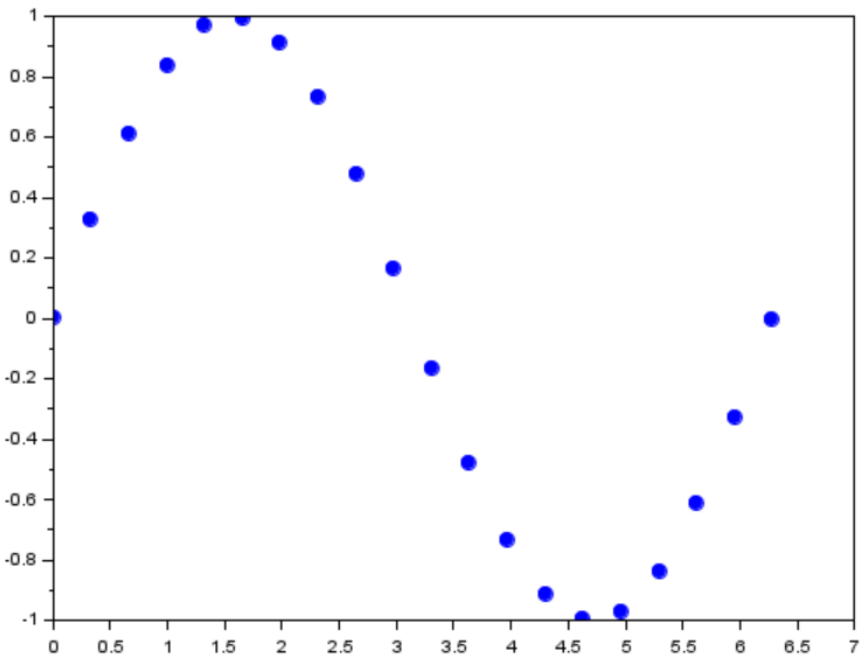

Рисунок 5 - моделирование RBF

Код программы сравнение значения функции:

```
// Определение интервала значений для точек
N = 20;
x = linspace(0, 2*%pi, N);

// Вычисление значений функции по заданной формуле
function y=myFunction(x)
    y = sin(x);
endfunction

formula_values = myFunction(x);

// Модель RBF-сети
input = x; // Входные данные RBF-сети
centers = linspace(0, 2*%pi, 10)'; // Центры RBF-сети (пример)
weights = ones(10, 1); // Веса RBF-сети (пример)
spread = 0.5; // Разброс RBF-сети (пример)

// Функция моделирования RBF-сети
function [output]=ann_rbf_run(input, centers, weights, spread)
    n = size(input, 1);
    k = size(centers, 1);

    distances = zeros(n, k);
    for i = 1:n
        for j = 1:k
            distances(i, j) = norm(input(i, :) - centers(j, :));
        end
    end

    activations = exp(-(distances.^2) / (2 * spread^2));

    output = activations * weights;
endfunction

// Получение значений функции с помощью модели RBF-сети
rbf_values = ann_rbf_run(input, centers, weights, spread);

// Вычисление значений функции в точках, не использовавшихся для обучения
x_test = linspace(0, 2*%pi, 100);
formula_values_test = myFunction(x_test);
rbf_values_test = ann_rbf_run(x_test, centers, weights, spread);

// Построение графиков
clf(); // Очистка текущего графика

// График значений функции в заданных точках N
plot(x, formula_values, 'b-', 'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 5,
'MarkerFaceColor', 'b');
hold on;
plot(x, rbf_values, 'r-', 'LineWidth', 2, 'Marker', 's', 'MarkerSize', 5,
'MarkerFaceColor', 'r');
xlabel('x');
ylabel('y');
legend('Формула', 'RBF-сеть');
title('Значения функции в заданных точках N');

// График значений функции в точках, не использовавшихся для обучения
clf(); // Очистка текущего графика
plot(x_test, formula_values_test, 'b-', 'LineWidth', 2, 'Marker', 'o',
'MarkerSize', 5, 'MarkerFaceColor', 'b');
hold on;
```

```
plot(x_test, rbf_values_test, 'r-', 'LineWidth', 2, 'Marker', 's',
'MarkerSize', 5, 'MarkerFaceColor', 'r');
xlabel('x');
ylabel('y');
legend('Формула', 'RBF-сеть');
title('Значения функции в точках, не использовавшихся для обучения');
```

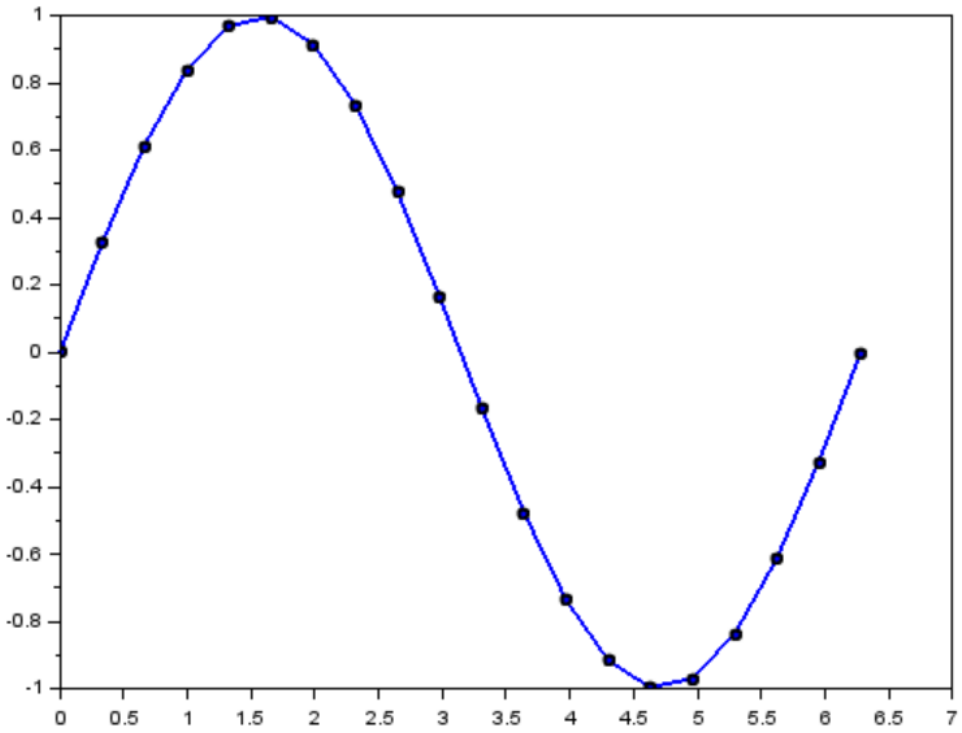

Рисунок 6 – график сравнения

Код программы средняя относительная ошибка аппроксимации функции:

```
// Генерация точек N
N = linspace(0, 2*%pi, 20)';

// Функция, которую вы хотите вычислить
function y=myFunction(x)
    y = sin(x);
endfunction

// Вычисление значений функции в точках N
formula_values = myFunction(N);

// Модель RBF-сети
input = N; // Входные данные RBF-сети
centers = linspace(0, 2*%pi, 10)'; // Центры RBF-сети (пример)
weights = ones(10, 1); // Веса RBF-сети (пример)
spread = 0.5; // Разброс RBF-сети (пример)

// Функция моделирования RBF-сети с визуализацией
function [output]=ann_rbf_run(input, centers, weights, spread)
    n = size(input, 1);
```

```
    k = size(centers, 1);

    distances = zeros(n, k);
    for i = 1:n
        for j = 1:k
            distances(i, j) = norm(input(i, :) - centers(j, :));
        end
    end

    activations = exp(-(distances.^2) / (2 * spread^2));

    output = activations * weights;
endfunction

// Получение значений функции с помощью модели RBF-сети
rbf_values = ann_rbf_run(input, centers, weights, spread);

// Вычисление средней относительной ошибки аппроксимации
relative_errors = zeros(size(formula_values));
for i = 1:length(formula_values)
    if abs(formula_values(i)) > 1e-10
        relative_errors(i) = abs(formula_values(i) - rbf_values(i)) /
abs(formula_values(i));
    else
        relative_errors(i) = 0; // Избегаем деления на ноль
    end
end

mean_relative_error = mean(relative_errors);

disp("Средняя относительная ошибка аппроксимации: " +
string(mean_relative_error));
```

```
->
 "Средняя относительная ошибка аппроксимации: 3.2904853"

->
```

Рисунок 7 – Средняя относительная ошибка аппроксимации

Вывод:

В ходе лабораторной работы была изучена архитектура RBF сетей. Были сформированы умения в области расчетов параметров RBF-сетей. Были приобретены навыки самостоятельного решения прикладной задачи с помощью нейросетей.