МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

**Лабораторная работа №3**

**по спецкурсу «Нейроинформатика»**

**Многослойные сети. Алгоритм обратного распространения ошибки**

Москва, 2022

### Цель работы

Исследование свойств многослойной нейронной сети прямого распространения и алгоритмов ее обучения, применение сети в задачах классификации и аппроксимации функции.

### Основные этапы работы

1. Использовать многослойную нейронную сеть для классификации точек в случае, когда классы не являются линейно разделимыми.
2. Использовать многослойную нейронную сеть для аппроксимации функции. Произвести обучение с помощью одного из методов первого порядка.
3. Использовать многослойную нейронную сеть для аппроксимации функции. Произвести обучение с помощью одного из методов второго порядка.

### 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

Этап 1

1. Заданы 3 линейно неразделимых класса. Точки, принадлежащие одному классу, лежат на алгебраической линии. Построить и обучить многослойную сеть прямого распространения, которая будет классифицировать точки заданной области.

Обучающий набор число классов . Сеть реализует отображение вида:

Эллипс: ;

Эллипс: ;

Эллипс: .

* 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. Для выбора точек рекомендуется использовать функцию *randperm*, с помощью которой получить псевдослучайную последовательность индексов вектора.



* 1. Множество точек, принадлежащее каждому классу, разделить на обучающее, контрольное, и тестовое подмножества с помощью функции *dividerand* в отношении 70%-20%-10%.
  2. Отобразить с помощью функции *plot* исходные множества точек для каждого из классов. Задать параметр *LineWidth* равным 2, подписать линии, задать сетку. С помощью *axis* задать границы для входного множества.



* 1. Соответствующие подмножества точек каждого класса объединить в обучающее, контрольное, и тестовое подмножества обучающей выборки. Обучающая выборка состоит из последовательного объединения полученных обучающего, контрольного, и тестового подмножеств.
  2. Создать сеть с помощью функции *feedforwardnet*. Сконфигурировать сеть (*configure*), указав диапазоны изменения для входного множества и эталонных выходов сети. Точки входного и выходного множеств лежат на отрезках и по каждой из координат соответственно.

Число нейронов скрытого слоя задать равным 20. Использовать активационные функцию *tansig* для скрытого и выходного слоев. Задать *RProp* в качестве алгоритма обучения.

* 1. Для разделения обучающего множества на подмножества использовать *net.divideFcn* =*'divideind'*. Также задать параметры:

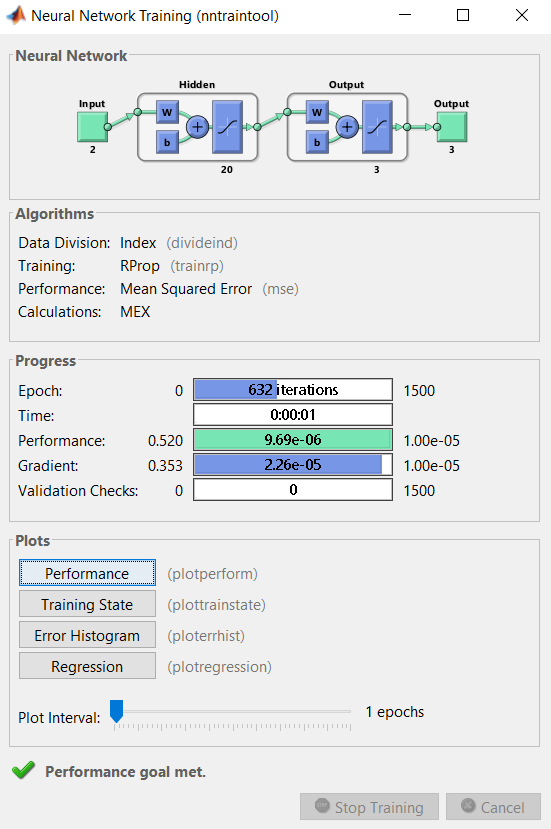
*net.divideParam.trainInd = 1 : trnInd;*

*net.divideParam.valInd = trnInd + 1 : tstInd;*

*net.divideParam.testInd = tstInd + 1 : proInd;*

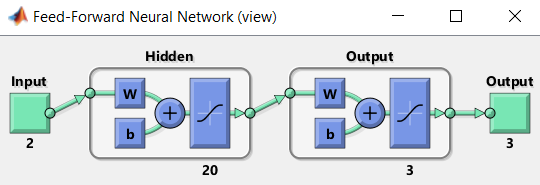
где *trnInd*, *tstInd*, *proInd* задают количество примеров в обучающем, контрольном, и тестовом подмножествах.

* 1. Инициализировать (*init*) весовые коэффициенты и смещения сети с помощью функции, заданной по умолчанию.
  2. Задать параметры обучения: число эпох обучения (*net.trainParam.epochs*) и число эпох, в течение которых может расти ошибка на контрольном подмножестве (*net.trainParam.max\_fail*), равными 1500, предельное значение критерия обучения (*net.trainParam.goal*) равным .
  3. Выполнить обучение сети с помощью функции *train*. Для обучения использовать обучающую выборку. Занести в отчет содержимое *Performance* и *Neural Network Training*.





* 1. Отразить структуру сети и проведенное обучение в отчете, заполнив таблицу 1.



|  |  |
| --- | --- |
| **Функция создания сети** | *feedforwardnet* |
| **Входной слой** | [2] |
| **Скрытый слой** | [20] |
| **Выходной слой** | [3] |
| **Активационные функции** | tansig, tansig |
| **Динамика** | минимизация *mse* |
| **Функция разделения обучающего множества** | *divideind* |
| **Число примеров в подмножествах** | *n\_train*=196  *n\_val*=56  *n\_test*=28 |
| **Метод обучения** | *train/trainrp* |
| **Параметры обучения** | *net.trainParam.epochs = 1500*  *net.trainParam.max\_fail = 1500*  *net.trainParam.goal = 1e-5* |
| **Метод инициализации сети** | *initlay* |
| **Критерий окончания обучения** | goal=1e-5  или  *trainParam.epochs = 1500* |
| **Причина окончания обучения** | достигнут критерий обучения |
| **Число эпох обучения** | 632 |

* 1. Рассчитать выход сети (*sim*) для обучающего подмножества. Преобразовать значения по правилу

Занести в отчёт количество правильно классифицированных точек.

Обучающие: 196/196

Проверочные: 56/56

Тестовые: 28/28

* 1. Провести аналогичные расчеты для контрольного и тестового подмножеств.
  2. Произвести классификацию точек области . Для этого задать сетку для указанной области с шагом . Рассчитать выход сети для всех узлов сетки.
  3. Выход сети для каждой точки задает ее принадлежность к трем классам. Закодировать принадлежности к классам различными цветами и занести полученное изображение в отчет.



Этапы 2 и 3

1. Задан обучающий набор . Построить и обучить двухслойную нейронную сеть прямого распространения, которая будет выполнять аппроксимацию функции вида

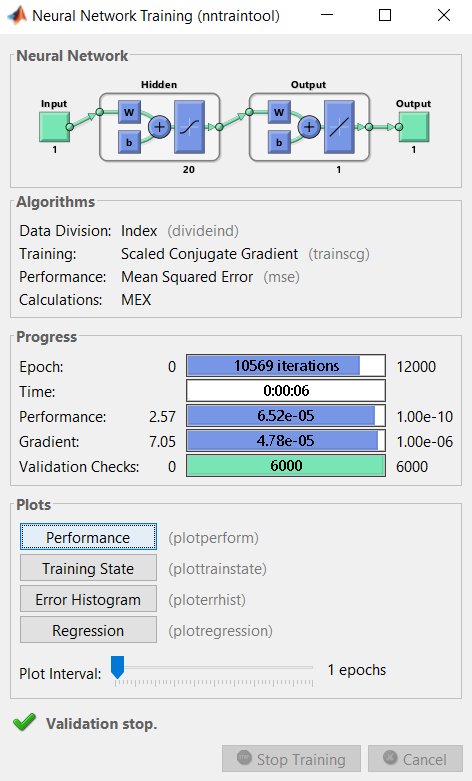
Для обучения использовать алгоритм, реализующий метод поиска экстремума функции многих переменных первого порядка. Функция и метод обучения определяются вариантом задания.

*trainscg*, *trainoss*

* 1. Создать сеть с помощью функции *feedforwardnet*. Сконфигурировать сеть под обучающее множество с помощью функции configure. Число нейронов скрытого слоя задать равным 10. Использовать активационные функции, заданные по умолчанию (*tansig*, *purelin*). Алгоритм обучения определяется вариантом задания.
  2. Для разделения обучающией выборки на обучающее, контрольное, и тестовое подмножества использовать функцию *divideind*. Выделить с конца временной последовательности 10% отсчетов на контрольное подмножество. Тестовое подмножество оставить пустым.

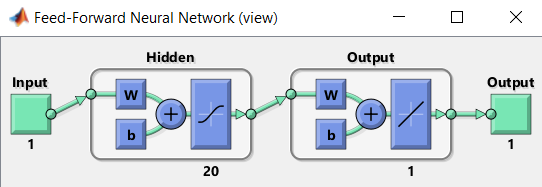
*net.divideParam.testInd = []*

* 1. Инициализировать сеть (*init*) с помощью функции, заданной по умолчанию
  2. Задать параметры обучения: значения параметров для некоторых методов обучения описаны выше, число эпох обучения (*net.trainParam.epochs*) и число эпох, в течение которых может расти ошибка на контрольном подмножестве (*net.trainParam.max\_fail*), равными 600, предельное значение критерия обучения (*net.trainParam.goal*) равным
  3. Выполнить обучение сети с помощью функции *train*. Если необходимо, то произвести обучение несколько раз. Если результаты неудовлетворительные или наблюдается переобучение, то изменить число нейронов в функции *feedforwardnet*, увеличить число эпох обучения или уменьшить предельное значение критерия обучения. Занести в отчет весовые коэффициенты и смещения для двух слоев. Занести в отчет окна *Performance* и *Neural Network Training*, если это возможно для данного метода обучения.





* 1. Отразить структуру сети и проведенное обучение в отчете, заполнив таблицу 1.



|  |  |
| --- | --- |
| **Функция создания сети** | *feedforwardnet* |
| **Входной слой** | [1] |
| **Скрытый слой** | [20] |
| **Выходной слой** | [1] |
| **Активационные функции** | tansig, purelin |
| **Динамика** | минимизация *mse* |
| **Функция разделения обучающего множества** | *divideind* |
| **Число примеров в подмножествах** | *n\_train*=181  *n\_val*=56  *n\_test*=0 |
| **Метод обучения** | *train/trainscg* |
| **Параметры обучения** | *net.trainParam.epochs = 12000;*  *net.trainParam.max\_fail = 6000;*  *net.trainParam.goal = 1e-10;* |
| **Метод инициализации сети** | *initlay* |
| **Критерий окончания обучения** | *trainParam.epochs = 12000*  или  *net.trainParam.goal = 1e-10* |
| **Причина окончания обучения** | достигнуто предельное количество эпох, на которых растёт ошибка на контрольной выборке |
| **Число эпох обучения** | 10569 |

* 1. 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.



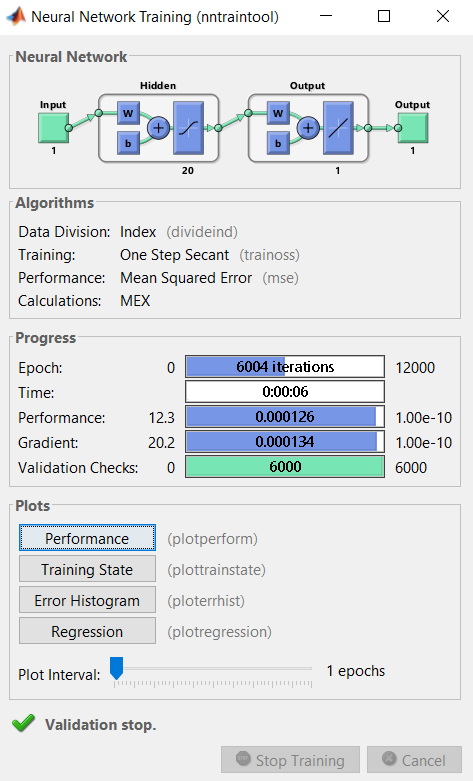
|  |  |
| --- | --- |
| *Обучающее подмножество* | |
| ***R* квадрат** | 0.999645 |
| ***MSE*** | 0.000182 |
| ***RMSE*** | 0.013475 |
| **Относительная СКО, %** | 0.680303 |
| ***MAE*** | 0.010996 |
| ***min absolute error*** | 0.000086 |
| ***max absolute error*** | 0.033018 |
| ***MAPE*, %** | 3.747539 |
| **Доля с ошибкой менее 5%, %** | 87.610619 |
| **Доля с ошибкой от 5% до 10%, %** | 7.079646 |
| **Доля с ошибкой от 10% до 20%, %** | 2.654867 |
| **Доля с ошибкой от 20% до 30%, %** | 0.884956 |
| **Доля с ошибкой более 30%, %** | 1.769912 |

* 1. Проделать тоже самое для контрольного подмножества.



1. Построить и обучить двухслойную нейронную сеть прямого распространения, которая будет выполнять аппроксимацию функции. Для обучения использовать алгоритм, реализующий метод оптимизации функций многих переменных второго порядка. Функция и метод обучения определяются вариантом задания.

Последовательности шагов для выполнения 2 и 3 этапов работы совпадают.





|  |  |
| --- | --- |
| **Функция создания сети** | *feedforwardnet* |
| **Входной слой** | [1] |
| **Скрытый слой** | [20] |
| **Выходной слой** | [1] |
| **Активационные функции** | tansig, purelin |
| **Динамика** | минимизация *mse* |
| **Функция разделения обучающего множества** | *divideind* |
| **Число примеров в подмножествах** | *n\_train*=316  *n\_val*=35  *n\_test*=0 |
| **Метод обучения** | *train/trainoss* |
| **Параметры обучения** | *net.trainParam.epochs = 12000;*  *net.trainParam.max\_fail = 6000;*  *net.trainParam.goal = 1e-10;* |
| **Метод инициализации сети** | *initlay* |
| **Критерий окончания обучения** | *trainParam.epochs = 12000*  или  *net.trainParam.goal = 1e-10* |
| **Причина окончания обучения** | Превышено число ошибок на контрольном подмножестве |
| **Число эпох обучения** | 6004 |



|  |  |
| --- | --- |
| *Обучающее подмножество* | |
| ***R* квадрат** | 0.979528 |
| ***MSE*** | 0.010563 |
| ***RMSE*** | 0.102778 |
| **Относительная СКО, %** | 4.874225 |
| ***MAE*** | 0.085082 |
| ***min absolute error*** | 0.000950 |
| ***max absolute error*** | 0.358382 |
| ***MAPE*, %** | 25.220360 |
| **Доля с ошибкой менее 5%, %** | 21.681416 |
| **Доля с ошибкой от 5% до 10%, %** | 24.778761 |
| **Доля с ошибкой от 10% до 20%, %** | 28.761062 |
| **Доля с ошибкой от 20% до 30%, %** | 12.389381 |
| **Доля с ошибкой более 30%, %** | 12.389381 |



|  |  |
| --- | --- |
| *Контрольное подмножество* | |
| ***R* квадрат** | 0.783188 |
| ***MSE*** | 0.022888 |
| ***RMSE*** | 0.151289 |
| **Относительная СКО, %** | 17.447969 |
| ***MAE*** | 0.140738 |
| ***min absolute error*** | 0.010348 |
| ***max absolute error*** | 0.217037 |
| ***MAPE*, %** | 53.563128 |
| **Доля с ошибкой менее 5%, %** | 0.000000 |
| **Доля с ошибкой от 5% до 10%, %** | 0.000000 |
| **Доля с ошибкой от 10% до 20%, %** | 8.000000 |
| **Доля с ошибкой от 20% до 30%, %** | 8.000000 |
| **Доля с ошибкой более 30%, %** | 84.000000 |

### 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);

CKO = 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, CKO, 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***

% ЛР3

% Вариант 10

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

%% построение множества точек

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, 0.7, 0.2, 0.1);

[g\_train, g\_val, g\_test] = dividerand(g, 0.7, 0.2, 0.1);

[b\_train, b\_val, b\_test] = dividerand(b, 0.7, 0.2, 0.1);

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\_val(1, :), r\_val(2, :), 'rV', ...

r\_test(1, :), r\_test(2, :), 'rs', ...

G(1, :), G(2, :), '-g', ...

g\_train(1, :), g\_train(2, :), 'og', ...

g\_val(1, :), g\_val(2, :), 'gV', ...

g\_test(1, :), g\_test(2, :), 'gs', ...

B(1, :), B(2, :), '-b', ...

b\_train(1, :), b\_train(2, :), 'ob', ...

b\_val(1, :), b\_val(2, :), 'bV', ...

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).MarkerEdgeColor = 'k';

p(4).MarkerFaceColor = 'c';

p(4).MarkerSize = 7;

p(5).LineWidth = 2;

p(6).MarkerEdgeColor = 'k';

p(6).MarkerFaceColor = 'g';

p(6).MarkerSize = 7;

p(7).MarkerEdgeColor = 'k';

p(7).MarkerFaceColor = 'c';

p(7).MarkerSize = 7;

p(8).MarkerEdgeColor = 'k';

p(8).MarkerFaceColor = 'c';

p(8).MarkerSize = 7;

p(9).LineWidth = 2;

p(10).MarkerEdgeColor = 'k';

p(10).MarkerFaceColor = 'b';

p(10).MarkerSize = 7;

p(11).MarkerEdgeColor = 'k';

p(11).MarkerFaceColor = 'c';

p(11).MarkerSize = 7;

p(12).MarkerEdgeColor = 'k';

p(12).MarkerFaceColor = 'c';

p(12).MarkerSize = 7;

axis([-1.5 1.5 -1.5 1.5]);

legend('R', 'R\_{train}', 'R\_{val}', 'R\_{test}', ...

'G', 'G\_{train}', 'G\_{val}', 'G\_{test}', ...

'B', 'B\_{train}', 'B\_{val}', 'B\_{test}');

grid on;

%% объединение в выборку с метками

X = [r\_train g\_train b\_train r\_val g\_val b\_val r\_test g\_test b\_test];

y = [[1; 0; 0] \* ones(1, length(r\_train)) [0; 1; 0] \* ones(1, length(g\_train)) [0; 0; 1] \* ones(1, length(b\_train)) ...

[1; 0; 0] \* ones(1, length(r\_val)) [0; 1; 0] \* ones(1, length(g\_val)) [0; 0; 1] \* ones(1, length(b\_val)) ...

[1; 0; 0] \* ones(1, length(r\_test)) [0; 1; 0] \* ones(1, length(g\_test)) [0; 0; 1] \* ones(1, length(b\_test))];

%% создание сети

net = feedforwardnet(20);

net = configure(net, X, y);

net.inputs{1}.range = [-1.5 1.5; ...

-1.5 1.5];

net.outputs{2}.range = [0 1; ...

0 1; ...

0 1];

net.layers{1}.transferFcn = 'tansig';

net.layers{2}.transferFcn = 'tansig';

net.trainFcn = 'trainrp';

net.divideFcn = 'divideind';

net.divideParam.trainInd = 1:n\_train;

net.divideParam.valInd = n\_train+1:n\_train+n\_val;

net.divideParam.testInd = n\_train+n\_val+1:n\_train+n\_val+n\_test;

net = init(net); % инициализаций весов

net.trainParam.epochs = 1500;

net.trainParam.max\_fail = 1500;

net.trainParam.goal = 1e-5;

%% обучение сети

net = train(net, X, y);

%% расчёт количества правильно классифицированных образцов

n\_right\_train = sum(sum((sim(net, X(:, 1:n\_train)) >= 0.5) == logical(y(:, 1:n\_train)), 1) == 3);

n\_right\_val = sum(sum((sim(net, X(:, n\_train+1:n\_train+n\_val)) >= 0.5) == logical(y(:, n\_train+1:n\_train+n\_val)), 1) == 3);

n\_right\_test = sum(sum((sim(net, X(:, n\_train+n\_val+1:n\_train+n\_val+n\_test)) >= 0.5) == logical(y(:,n\_train+n\_val+1:n\_train+n\_val+n\_test)), 1) == 3);

fprintf('Обучающие: %d/%d\nПроверочные: %d/%d\nТестовые: %d/%d\n', ...

n\_right\_train, n\_train, ...

n\_right\_val, n\_val, ...

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]));

%% Создание обучающей выборки и сети

f = @(t) sin(t .^ 2 - 7 \* t);

t = 0:0.025:5;

X = t;

y = f(t);

net = feedforwardnet(20);

net = configure(net, X, y);

net.layers{1}.transferFcn = 'tansig';

net.layers{2}.transferFcn = 'purelin';

net.trainFcn = 'trainoss'; % 'trainscg'/'trainoss'

n\_train = ceil(length(X) \* 0.9);

net.divideFcn = 'divideind';

net.divideParam.trainInd = 1:n\_train;

net.divideParam.valInd = n\_train+1:length(X);

net.divideParam.testInd = [];

net = init(net); % инициализаций весов

net.trainParam.epochs = 2 \* 6000;

net.trainParam.max\_fail = 6000;

net.trainParam.goal = 1e-10;

%% Обучение

net = train(net, X, y);

%% Метрики и графики для обучающего подмножества

disp(accuracy(sim(net, X(1:n\_train)), y(1:n\_train)));

p = plot(X(1:n\_train), y(1:n\_train), ...

X(1:n\_train), sim(net, X(1:n\_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 контрольного подмножества

disp(accuracy(sim(net, X(n\_train+1:length(X))), y(n\_train+1:length(X))));

p = plot(X(n\_train+1:length(X)), y(n\_train+1:length(X)), ...

X(n\_train+1:length(X)), sim(net, X(n\_train+1:length(X))), '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 многослойной нейронной сети прямого распространения и алгоритмов её обучения. Было продемонстрировано применение сети в задачах классификации и аппроксимации функции.