**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 3**

по курсу «Нейроинформатика»

Тема: Многослойные сети. Алгоритм обратного распространения ошибки.

Студент: Дубинин А. О.

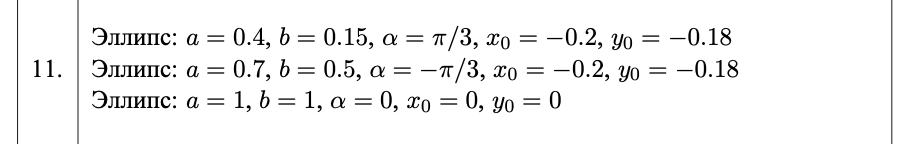
Группа: 8О-407Б-17

Преподаватель: Аносова Н.П.

Оценка:

Москва, 2021

Вариант 11





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.

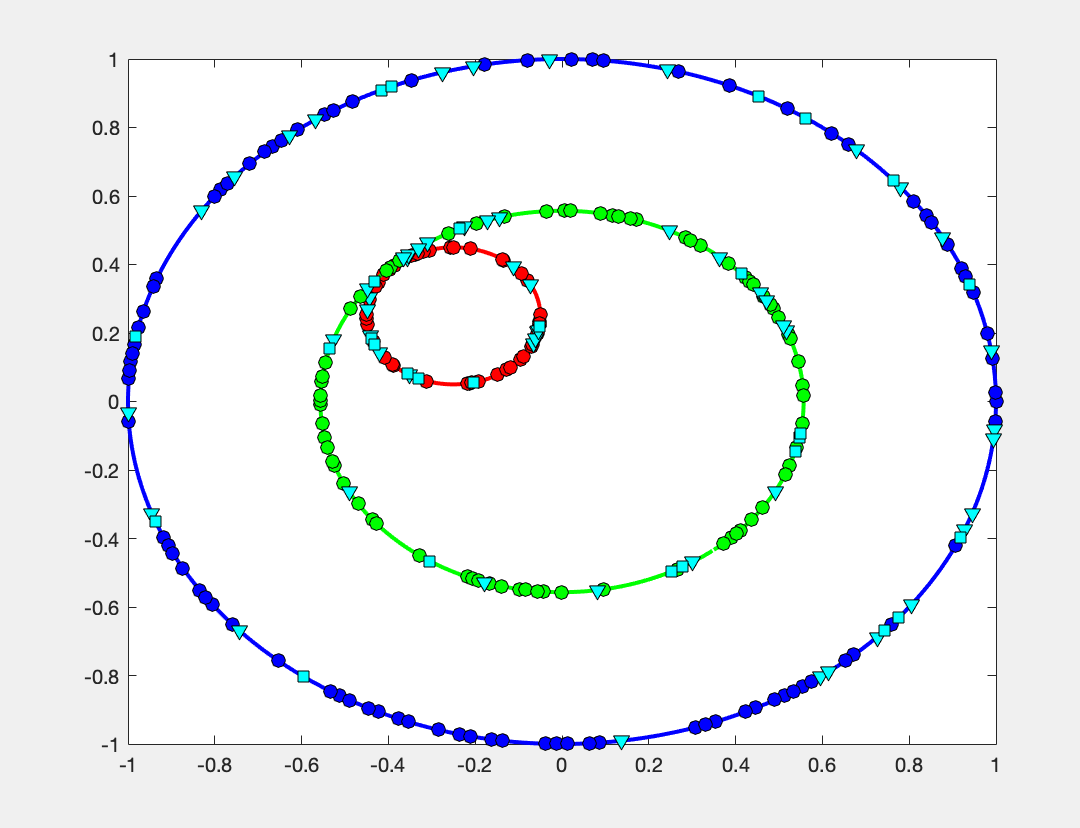
1.1 В соответствии с вариантом задания для каждой линии сгенерировать множество точек. Далее для первого класса выбрать из исходного множества случайным образом 60 точек. Для второго и третьего классов 100 и 120 точек соответственно. Для выбора точек рекомендуется использовать функцию randperm, с помощью которой получить псевдослучайную последовательность индексов вектора.

*%1.1 Генерация множества точек для каждой линии*  
  
n = length(X1);  
*%Случайные точки 1-го класса (60 шт)*D1 = randperm(n);  
n1 = 60;  
D1 = D1(1:n1);  
display(D1);  
K1 = [ones(1, n1); 0\*ones(1, n1); 0\*ones(1, n1)];  
  
*%Случайные точки 2-го класса (100 шт)*D2 = randperm(n);  
n2 = 100;  
D2 = D2(1:n2);  
display(D2);  
K2 = [0\*ones(1, n2); ones(1, n2); 0\*ones(1, n2)];  
  
*%Случайные точки 3-го класса (120 шт)*D3 = randperm(n);  
n3 = 120;  
D3 = D3(1:120);  
display(D3);  
K3 = [0\*ones(1, n3); 0\*ones(1, n3); ones(1, n3)];

1.2 Множество точек, принадлежащее каждому классу, разделить на обучающее, контрольное, и тестовое подмножества с помощью функции dividerand в отношении 70%- 20%-10%.

*%1.2 Разделение точек на обучающее, контрольное,   
%и тестовое подмножества в отношении 70%-20%-10%*[trainInd1, valInd1, testInd1] = dividerand(length(D1), 0.7, 0.2, 0.1);  
[trainInd2, valInd2, testInd2] = dividerand(length(D2), 0.7, 0.2, 0.1);  
[trainInd3, valInd3, testInd3] = dividerand(length(D3), 0.7, 0.2, 0.1);

1.3 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



1.4 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.

*%1.4 Создание обучающей выборки*trainset = [trainInd1, trainInd2+60, trainInd3+160,...  
 valInd1, valInd2+60, valInd3+160, ...  
 testInd1, testInd2+60, testInd3+160];

1.5 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 [−1.2,1.2] и [0,1] 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.

*%1.5 Создание сети*net = feedforwardnet(20);  
net = configure(net, [-1.2 1.2; 0 1]);  
net.layers{:}.transferFcn = **'tansig'**;  
net.trainFcn = **'trainrp'**;

1.6 Для разделения обучающего множества на подмножества использовать net.divideFcn = ‘divideind’ .

*%1.6 Разделение обучающего множества на подмножества*net.divideFcn = **'divideind'**;  
  
trnInd = length(trainInd1) + length(trainInd2) + length(trainInd3);  
tstInd = length(valInd1) + length(valInd2) + length(valInd3);  
proInd = length(testInd1) + length(testInd2) + length(testInd3);  
  
net.divideParam.trainInd = 1:trnInd;  
net.divideParam.valInd = (1:tstInd) + trnInd;  
net.divideParam.testInd = (1:proInd) + (tstInd + trnInd);

1.7 Инициализировать (init) весовые коэффициенты и смещения сети с помощью функции, заданной по умолчанию.

*%1.7 Инициализация*net = init(net);

1.8 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 10 −5 .

*%1.8 Задание параметров обучения*net.trainParam.epochs = 3000;  
net.trainParam.max\_fail = 1500;  
net.trainParam.goal = 0.00001;

1.9 Выполнить обучение сети с помощью функции train. Для обучения использовать обучающую выборку. Занести в отчет содержимое Performance и Neural Network Training.

*%1.9 Обучение сети*D = [D1, D2+length(X1), D3+length(X1)+length(X2)];  
X = [X1, X2, X3];  
Y = [Y1, Y2, Y3];  
K = [K1, K2, K3];

[net, tr] = train(net, [X(D(trainset)); Y(D(trainset))], K(:,trainset));

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

*%1.10 Отображение сети*display(net)

net =

Neural Network

name: 'Feed-Forward Neural Network'

userdata: (your custom info)

dimensions:

numInputs: 1

numLayers: 2

numOutputs: 1

numInputDelays: 0

numLayerDelays: 0

numFeedbackDelays: 0

numWeightElements: 123

sampleTime: 1

connections:

biasConnect: [1; 1]

inputConnect: [1; 0]

layerConnect: [0 0; 1 0]

outputConnect: [0 1]

subobjects:

input: Equivalent to inputs{1}

output: Equivalent to outputs{2}

inputs: {1x1 cell array of 1 input}

layers: {2x1 cell array of 2 layers}

outputs: {1x2 cell array of 1 output}

biases: {2x1 cell array of 2 biases}

inputWeights: {2x1 cell array of 1 weight}

layerWeights: {2x2 cell array of 1 weight}

functions:

adaptFcn: 'adaptwb'

adaptParam: (none)

derivFcn: 'defaultderiv'

divideFcn: 'divideind'

divideParam: .trainInd, .valInd, .testInd

divideMode: 'sample'

initFcn: 'initlay'

performFcn: 'mse'

performParam: .regularization, .normalization

plotFcns: {'plotperform', 'plottrainstate', 'ploterrhist',

'plotregression'}

plotParams: {1x4 cell array of 4 params}

trainFcn: 'trainrp'

trainParam: .showWindow, .showCommandLine, .show, .epochs,

.time, .goal, .min\_grad, .max\_fail, .delta0,

.delt\_inc, .delt\_dec, .deltamax

weight and bias values:

IW: {2x1 cell} containing 1 input weight matrix

LW: {2x2 cell} containing 1 layer weight matrix

b: {2x1 cell} containing 2 bias vectors

methods:

adapt: Learn while in continuous use

configure: Configure inputs & outputs

gensim: Generate Simulink model

init: Initialize weights & biases

perform: Calculate performance

sim: Evaluate network outputs given inputs

train: Train network with examples

view: View diagram

unconfigure: Unconfigure inputs & outputs

1.11 Рассчитать выход сети (sim) для обучающего подмножества. Занести в отчет количество правильно классифицированных точек.

*%1.11 Выход сети для обучающего, контрольного и тестового подмножеств, кол-во совпадений*trainInd = [trainInd1, trainInd2+60, trainInd3+160];  
valInd = [valInd1, valInd2+60, valInd3+160];  
testInd = [testInd1, testInd2+60, testInd3+160];  
  
*%Для обучающего*A = net([X(D(trainInd)); Y(D(trainInd))]);  
nA = A >= 0.5;  
  
fprintf(**'Размер обучающей выборки: %d\n'**,length(trainInd));  
fprintf(**'Количество совпадений: %d\n\n'**,sum((sum(K(:,trainInd) == nA))==3));

1.12 Провести аналогичные расчеты для контрольного и тестового подмножеств.

*%1.12  
%Для контрольного*A = net([X(D(valInd)); Y(D(valInd))]);  
nA = A >= 0.5;  
  
fprintf(**'Размер контрольной выборки: %d\n'**,length(valInd));  
fprintf(**'Количество совпадений: %d\n\n'**,sum((sum(K(:,valInd) == nA))==3));  
  
*%Для тестового*A = net([X(D(testInd)); Y(D(testInd))]);  
nA = A >= 0.5;  
  
fprintf(**'Размер тестовой выборки: %d\n'**,length(testInd));  
fprintf(**'Количество совпадений: %d\n\n'**,sum((sum(K(:,testInd) == nA))==3));

Размер обучающей выборки: 196

Количество совпадений: 196

Размер контрольной выборки: 56

Количество совпадений: 56

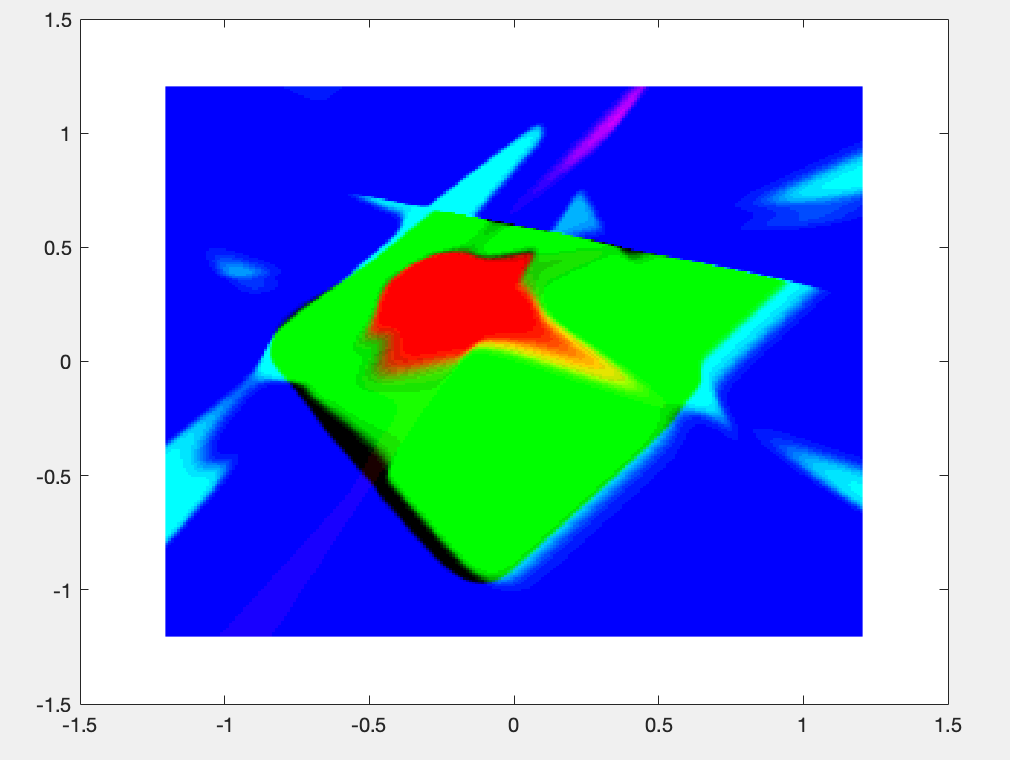
Размер тестовой выборки: 28

Количество совпадений: 26

1.13 Произвести классификацию точек области [−1.2,1.2]×[−1.2,1.2]. Для этого задать сетку для указанной области с шагом h = 0.025. Рассчитать выход сети для всех узлов сетки.

*%1.13 Классификация точек области [-1.2,1.2]x[-1.2,1.2]*M = -1.2:0.010:1.2;  
[gX, gY] = meshgrid(M);  
  
A = net([gX(:)';gY(:)']);

1.14 Выход сети для каждой точки задает ее принадлежность к трем классам. Закодировать принадлежности к классам различными цветами и занести полученное изображение в отчет.



Этап 2

2.1 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.

*% 2.1  
% Создаем сеть и конфигурируем под обучающее множество*net = feedforwardnet(20, **'traingdx'**); *% Градиентный спуск с адапт шагом*net.trainParam.lr = 0.05;  
net.trainParam.lr\_inc = 1.05;  
net.trainParam.mc = 0.9;  
  
net = configure(net, X, Y);

2.2 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.

*% 2.2  
% Разбиваем выборку на обучающее, контрольное и тестовое подмножества*trainInd = 1 : floor(n \* 0.9);  
valInd = floor(n \* 0.9) + 1 : n;  
testInd = [];  
net.divideFcn = **'divideind'**;  
net.divideParam.trainInd = trainInd;  
net.divideParam.valInd = valInd;  
net.divideParam.testInd = testInd;

2.3 Инициализировать сеть (init) с помощью функции, заданной по умолчанию.

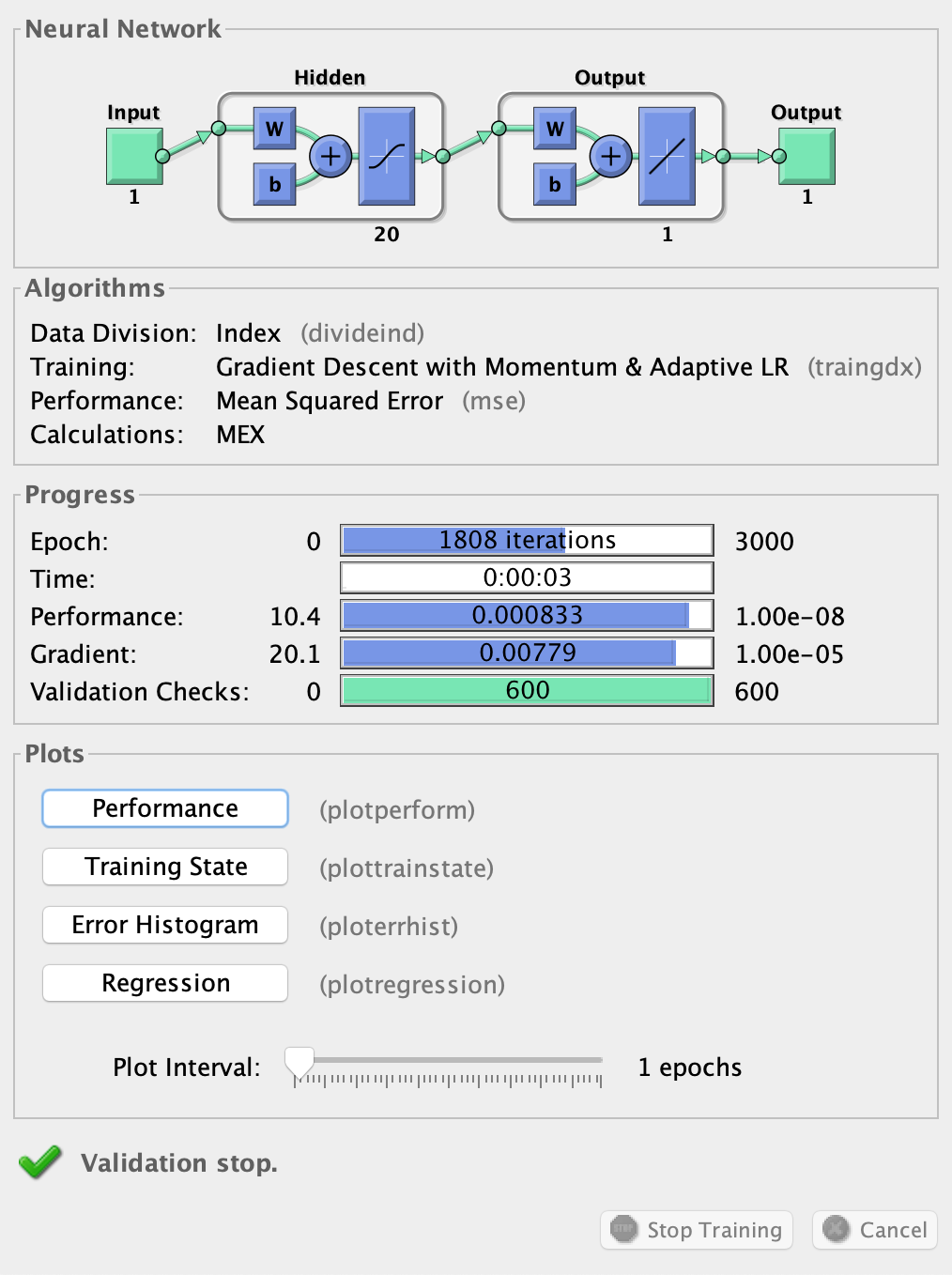
*% 2.3*net = init(net);

2.4 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 10^−8.

*% 2.4*net.trainParam.epochs = 3000;  
net.trainParam.max\_fail = 600;  
net.trainParam.goal = 1.0e-8;

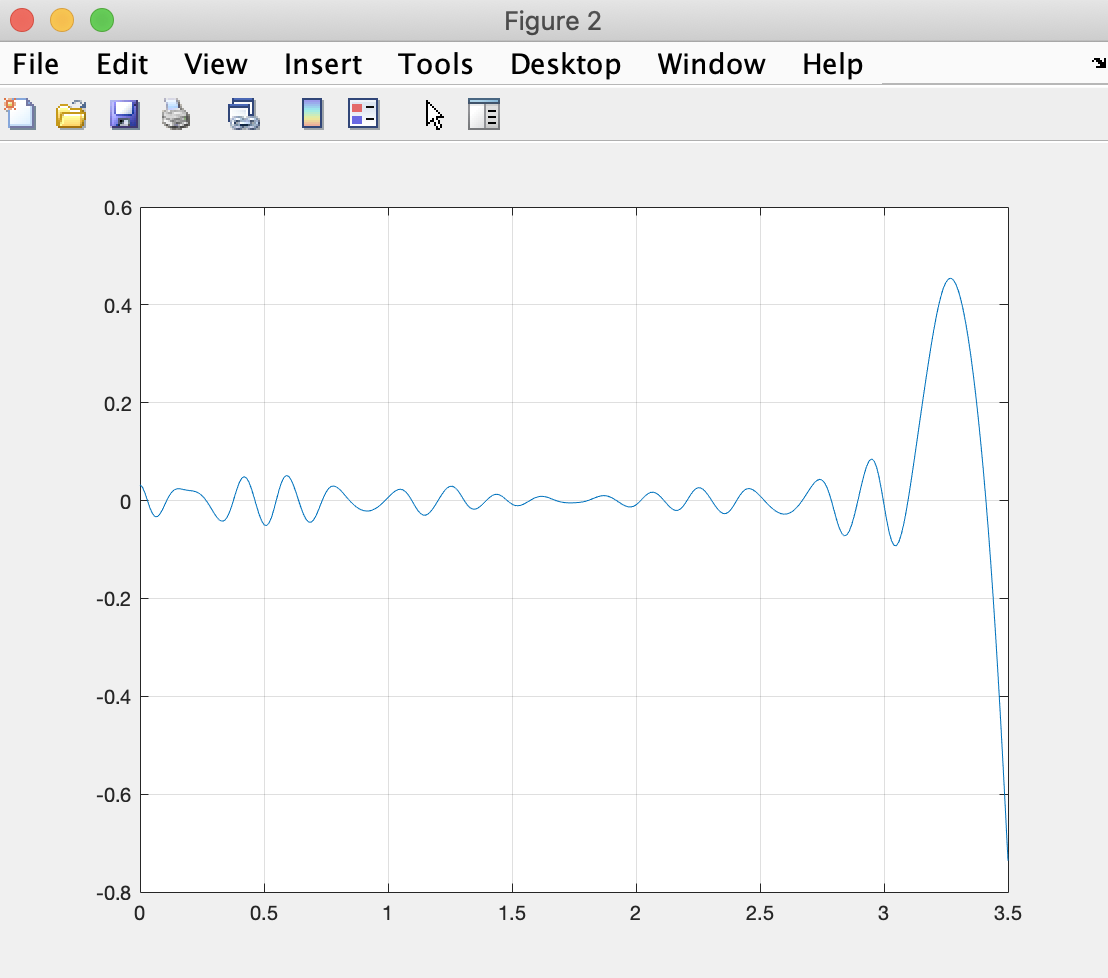
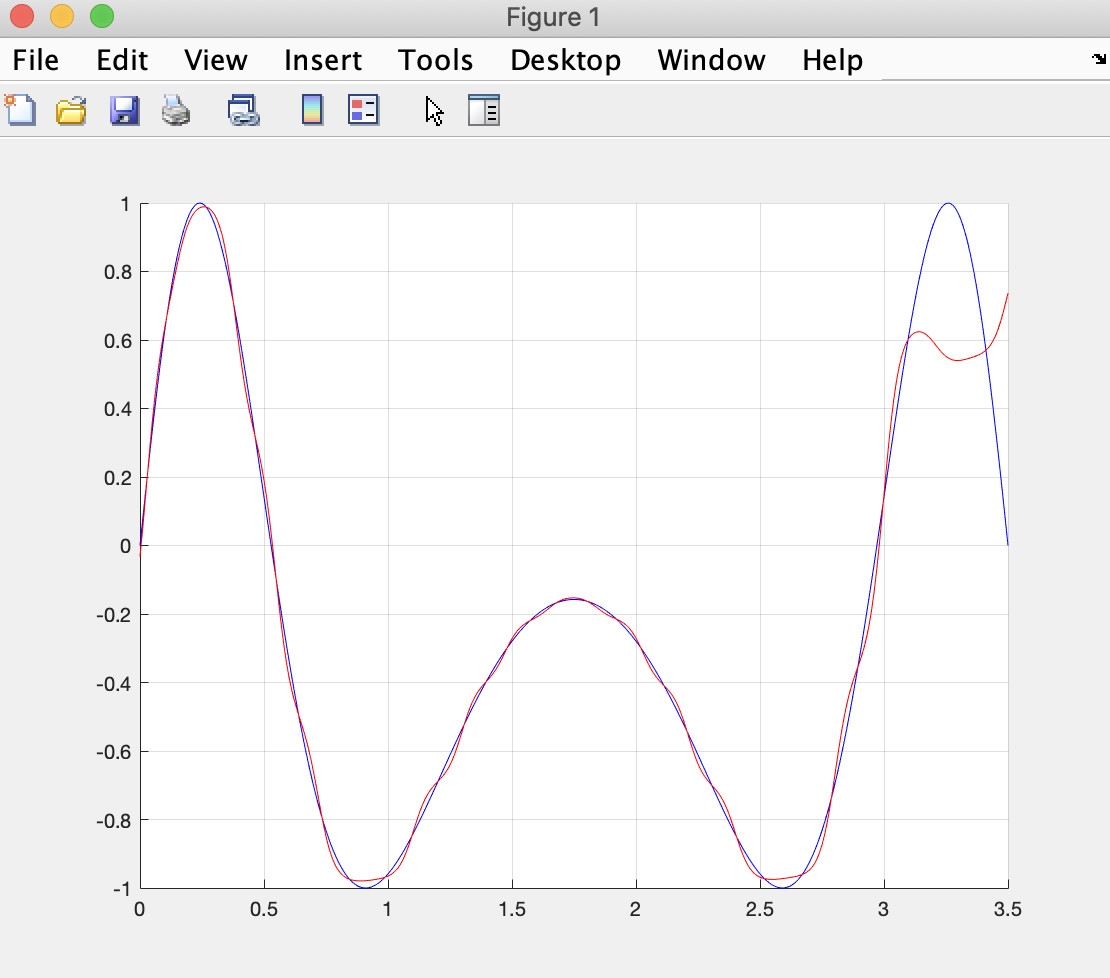
2.5 Выполнить обучение сети с помощью функции train. Если необходимо, то произвести обучение несколько раз. Если результаты неудовлетворительные или наблюдается переобучение, то изменить число нейронов в функции feedforwardnet, увеличить число эпох обучения или уменьшить предельное значение критерия 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.

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



2.7 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.

2.8 Проделать тоже самое для контрольного подмножества.



Этап 3

3.1 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.

*% 3.1  
% Создаем сеть и конфигурируем под обучающее множество*net = feedforwardnet(20, **'trainoss'**); *% одношаговый метод секущих*net = configure(net, X, Y);

3.2 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.

*% 3.2  
% Разбиваем выборку на обучающее, контрольное и тестовое подмножества*trainInd = 1 : floor(n \* 0.9);  
valInd = floor(n \* 0.9) + 1 : n;  
testInd = [];  
net.divideFcn = **'divideind'**;  
net.divideParam.trainInd = trainInd;  
net.divideParam.valInd = valInd;  
net.divideParam.testInd = testInd;

3.3 Инициализировать сеть (init) с помощью функции, заданной по умолчанию.

*% 3.3*net = init(net);

3.4 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 10^−8.

*% 3.4*net.trainParam.epochs = 3000;  
net.trainParam.max\_fail = 600;  
net.trainParam.goal = 1.0e-8;

3.5 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.

*% 3.5  
% Обучаем сеть*net = train(net, X, Y);

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

3.7 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.

*% 3.7*R = sim(net, X);  
  
sqrt(mse(Y(trainInd) - R(trainInd)))  
  
sqrt(mse(Y(valInd) - R(valInd)))  
  
figure;  
hold on;  
plot(X, Y, **'-b'**);  
plot(X, R, **'-r'**);  
grid on;  
  
figure;  
plot(X, Y - R);  
grid on;

3.8 Проделать тоже самое для контрольного подмножества.

