**Лабораторная работа № 8**

«Динамические сети».

Постановка задачи.

Целью работы является исследование свойств некоторых динамических нейронных сетей,

алгоритмов обучения, а также применение сетей в задачах аппроксимации функций и распознавания динамических образов.

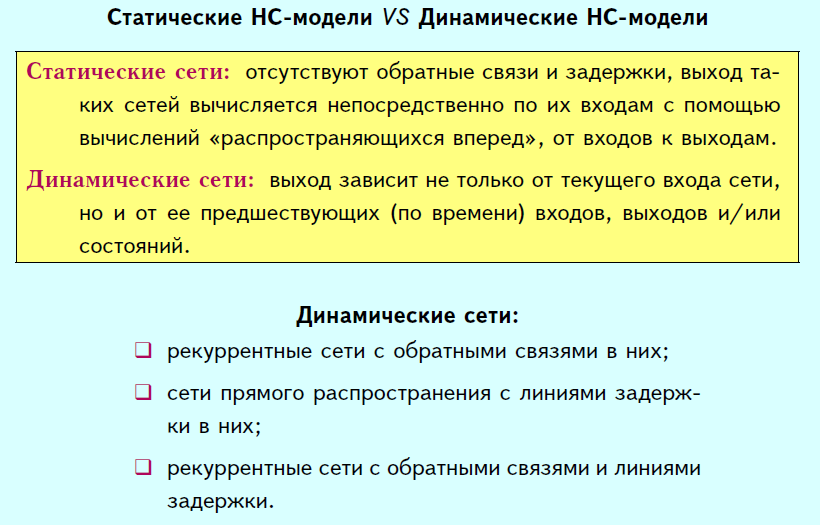
Ход работы.

1. Использовать сеть прямого распространения с запаздыванием для предсказания значений временного ряда и выполнения многошагового прогноза.

2. Использовать сеть прямого распространения с распределенным запаздыванием для распознавания динамических образов.

3. Использовать нелинейную авторегрессионную сеть с внешними входами для аппроксимации траектории динамической системы и выполнения многошагового прогноза.

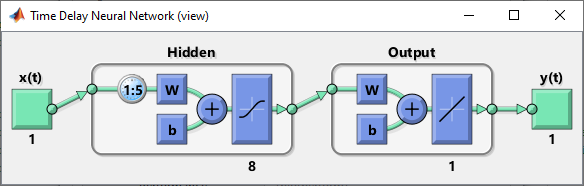
Теория.

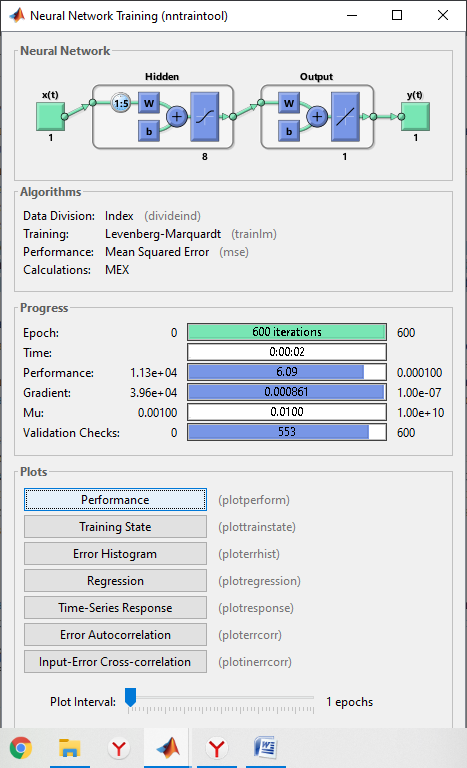


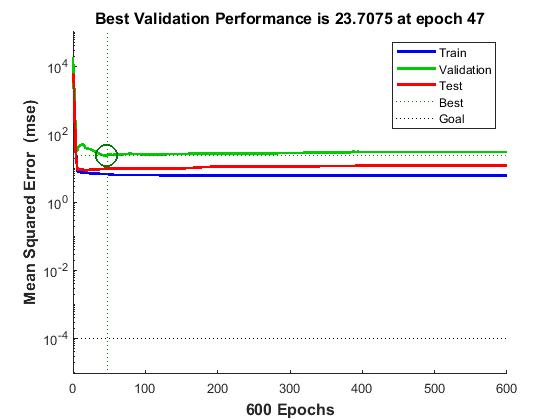
Входные данные и результаты.

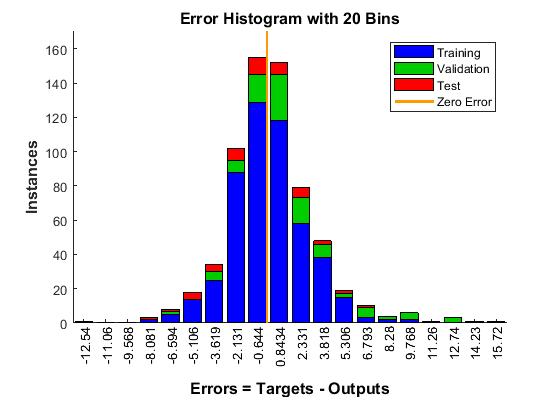
Вариант № 15.

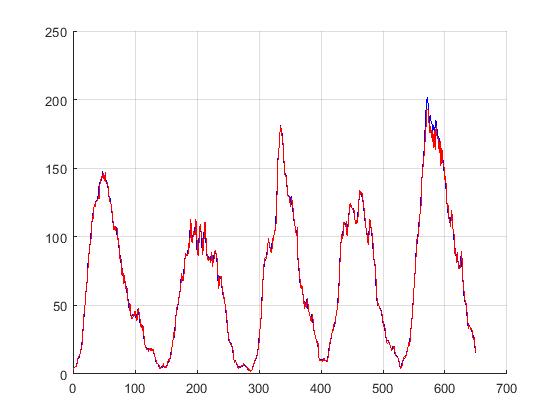
Первое задание



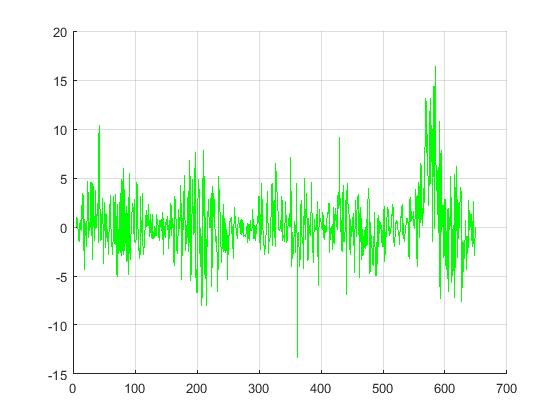


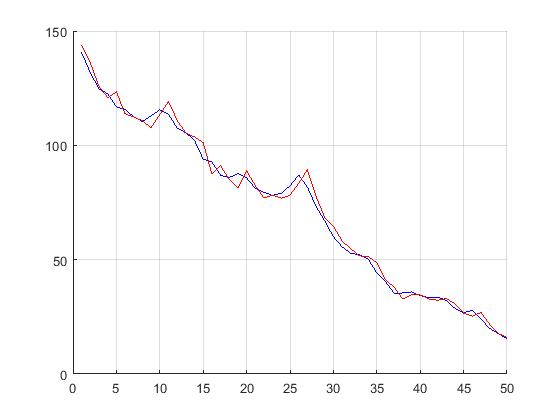




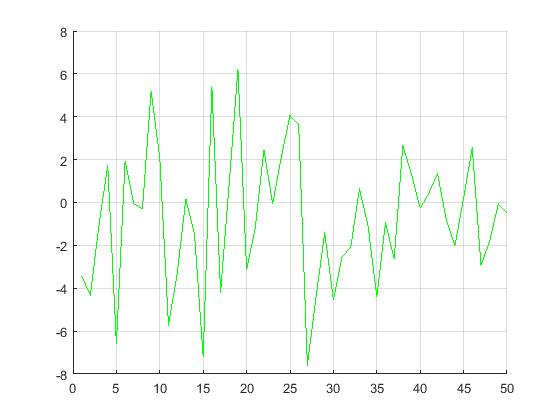


**Error plot**

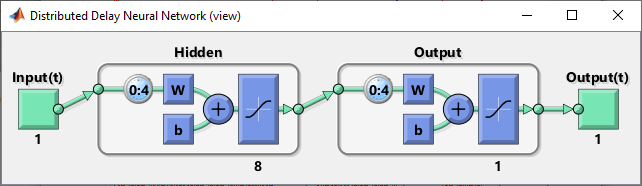


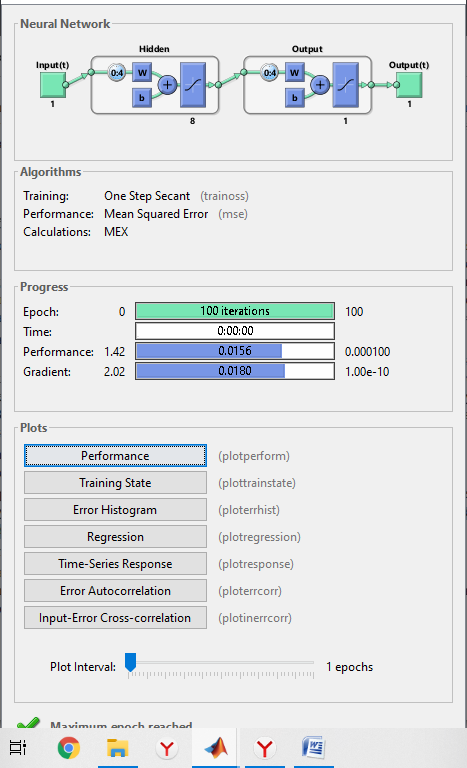


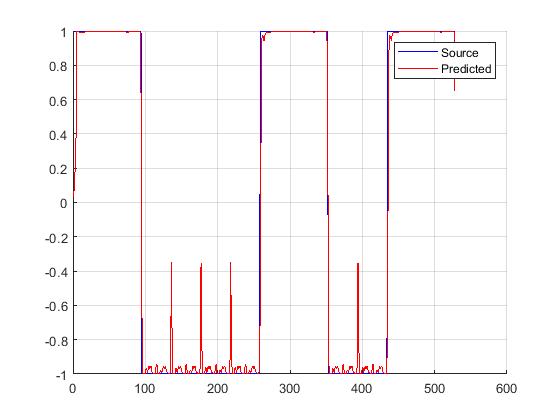
**Error plot**

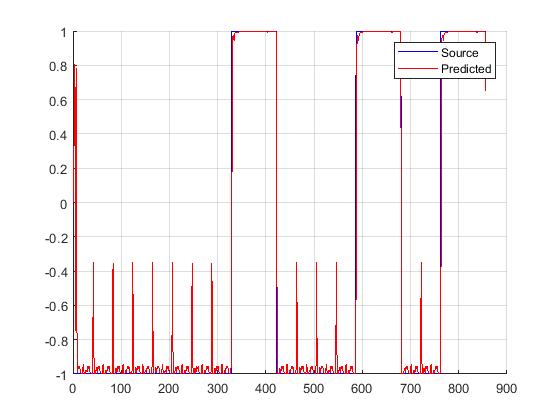


Второе задание

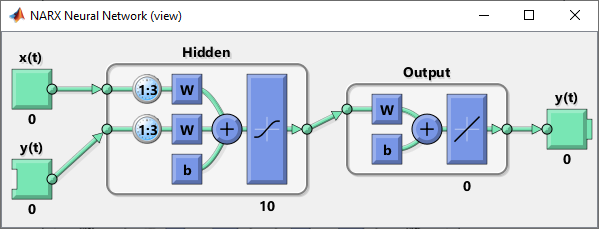


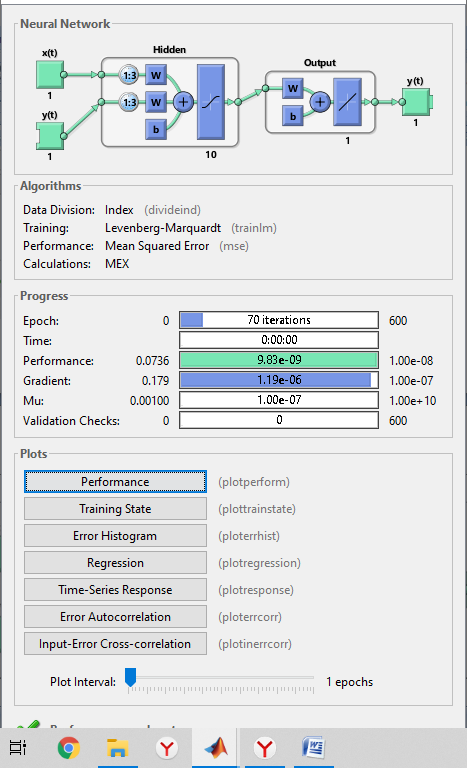


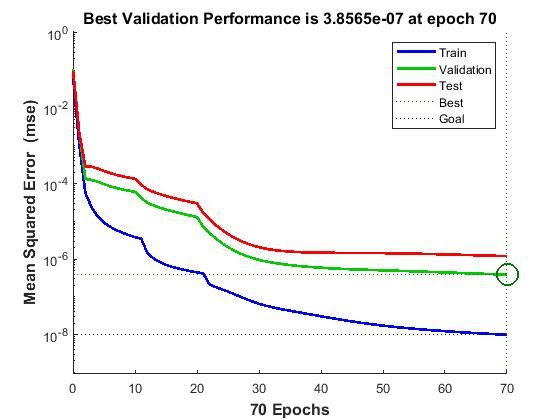


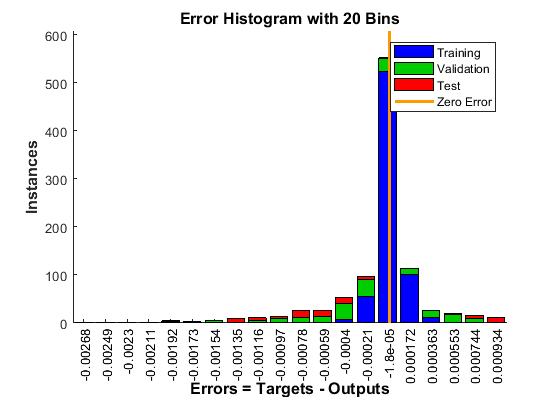


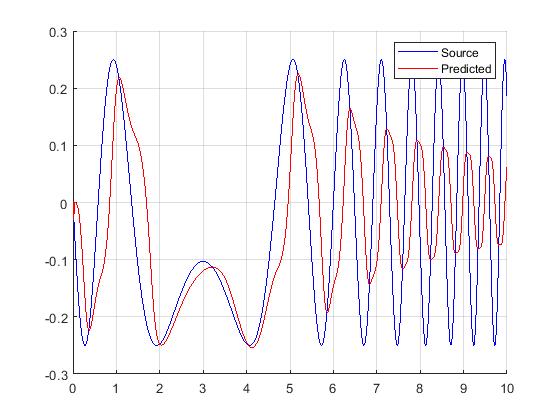
Третье задание



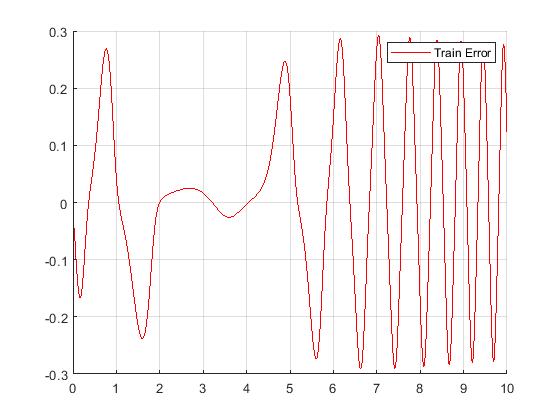


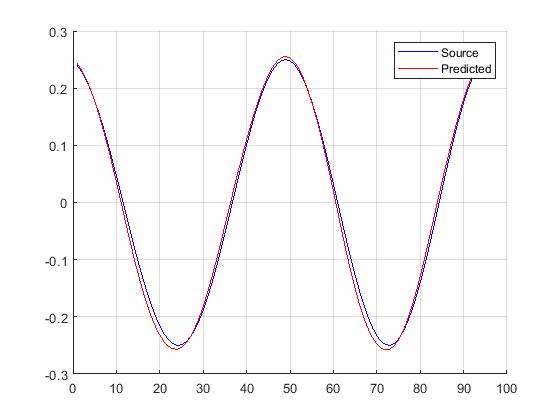




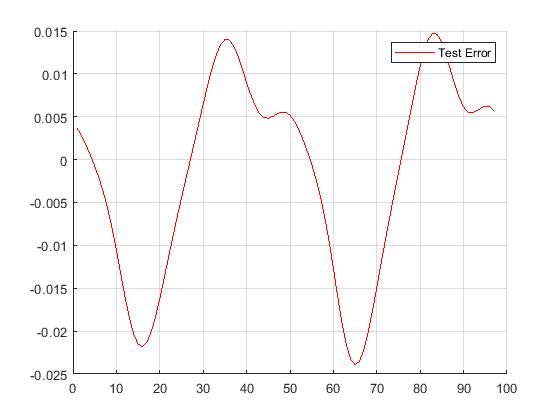


**Error plot**





**Error plot**



Исходный код.

clear;

clc;

%In the TXT and CSV files, the missing values are marked by -1 (valid Sunspot Number are always positive).

%Contents:

    %Column 1-3: Gregorian calendar date

        %- Year

        %- Month

        %- Day

    %Column 4: Date in fraction of year

    %Column 5: Daily total sunspot number. A value of -1 indicates that no number is available for that day (missing value).

    %Column 6: Daily standard deviation of the input sunspot numbers from individual stations.

    %Column 7: Number of observations used to compute the daily value.

data = load('mySunData.txt');

nums = data(:, 4);

x = smooth(nums, 12);

%x = con2seq(x(1:500 + 100 + 50)');

x = con2seq(x(1:650)');

D = 5;

net = timedelaynet(1:D, 8, 'trainlm');

net.layers{1}.transferFcn = 'tansig';

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

net.divideFcn = 'divideind';

net.divideParam.trainInd = 1 : 500;

net.divideParam.valInd = 501 : 600;

net.divideParam.testInd = 601 : 650;

net = configure(net, x, x);

net = init(net);

net.trainParam.epochs = 600;

net.trainParam.max\_fail = 600;

net.trainParam.goal = 10e-5;

view(net);

%Prepare input and target time series data for network simulation or training

[Xs, Xi, Ai, Ts] = preparets(net, x, x);

% Xs Shifted inputs

% Xi Initial input delay states

% Ai Initial layer delay states

% Ts Shifted targets

net = train(net, Xs, Ts, Xi, Ai);

Y = sim(net, Xs, Xi);

figure;

hold on;

grid on;

plot(cell2mat(x), '-b');

plot([cell2mat(Xi), cell2mat(Y)], '-r');

figure;

hold on;

grid on;

error = cell2mat(x)- [cell2mat(Xi), cell2mat(Y)];

plot(error, '-g');

xCheck = cell2mat(x);

yCheck = cell2mat(Y);

figure;

hold on;

grid on;

plot(xCheck(601 : 650), '-b');

plot(yCheck(601 - D : 650 - D), '-r');

figure;

hold on;

grid on;

error = xCheck(601 : 650)- yCheck(601 - D : 650 - D);

plot(error, '-g');

clear;

clc;

k = 0 : 0.025 : 1;

p1 = sin(4 \* pi \* k);

t1 = -ones(size(k));

k = 0.92 : 0.025 : 3.25;

g = @(k) cos( -2 \* k .^ 2 + 7 \* k);

p2 = g(k);

t2 = ones(size(k));

R = {0; 4; 2};

P = [repmat(p1, 1, R{1}), p2, repmat(p1, 1, R{2}), p2, repmat(p1, 1, R{3}), p2];

T = [repmat(t1, 1, R{1}), t2, repmat(t1, 1, R{2}), t2, repmat(t1, 1, R{3}), t2];

Pseq = con2seq(P);

Tseq = con2seq(T);

net = distdelaynet({0 : 4, 0 : 4}, 8, 'trainoss');

net.layers{1}.transferFcn = 'tansig';

net.layers{2}.transferFcn = 'tansig';

net.divideFcn = '';

net = configure(net, Pseq, Tseq);

view(net);

net.trainParam.epochs = 100;

net.trainParam.goal = 10e-5;

[Xs, Xi, Ai, Ts] = preparets(net, Pseq, Tseq);

net = train(net, Xs, Ts, Xi, Ai);

Y = sim(net, Xs, Xi);

figure;

hold on;

grid on;

plot(cell2mat(Tseq), '-b');

plot([cell2mat(Xi) cell2mat(Y)], '-r');

legend('Source', 'Predicted');

Yc = zeros(1, numel(Xi) + numel(Y));

for i = 1 : numel(Xi)

    if Xi{i} >= 0

        Yc(i) = 1;

    else

        Yc(i) = -1;

    end

end

for i = numel(Xi) + 1 : numel(Y)

    if Y{i} >= 0

        Yc(i) = 1;

    else

        Yc(i) = -1;

    end

end

display( nnz(Yc == cell2mat(Tseq)) / length(Tseq) )

R = {8; 4; 2};

P = [repmat(p1, 1, R{1}), p2, repmat(p1, 1, R{2}), p2, repmat(p1, 1, R{3}), p2];

T = [repmat(t1, 1, R{1}), t2, repmat(t1, 1, R{2}), t2, repmat(t1, 1, R{3}), t2];

Pseq = con2seq(P);

Tseq = con2seq(T);

[Xs, Xi, Ai, Ts] = preparets(net, Pseq, Tseq);

Y = sim(net, Xs, Xi);

figure;

hold on;

grid on;

plot(cell2mat(Tseq), '-b');

plot([cell2mat(Xi) cell2mat(Y)], '-r');

legend('Source', 'Predicted');

Yc = zeros(1, numel(Xi) + numel(Y));

for i = 1 : numel(Xi)

    if Xi{i} >= 0

        Yc(i) = 1;

    else

        Yc(i) = -1;

    end

end

for i = numel(Xi) + 1 : numel(Y)

    if Y{i} >= 0

        Yc(i) = 1;

    else

        Yc(i) = -1;

    end

end

display( nnz(Yc == cell2mat(Tseq)) / length(Tseq) )

clear;

clc;

varFunc = @(k)sin(k.^2-6.\*k - 2 \* pi)/4;

commonFunc = @(y, u)y ./ (1 + y.^2) + u.^3;

tStart = 0;

tEnd = 10;

h = 0.01;

n = 1 + (tEnd - tStart) / h;

u = zeros(1, n);

y = zeros(1, n);

u(1) = varFunc(0);

for i = 2 : n

    t = tStart + (i - 1) \* h;

    y(i) = commonFunc(y(i - 1), u(i - 1));

    u(i) = varFunc(t);

end

x = u;

D = 3;

trainInd = 1 : 700;

valInd = 701 : 900;

testInd = 901 : 997;

net = narxnet(1 : 3, 1 : 3, 10);

net.trainFcn = 'trainlm';

net.layers{1}.transferFcn = 'tansig';

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

net.divideFcn = 'divideind';

net.divideParam.trainInd = trainInd;

net.divideParam.valInd = valInd;

net.divideParam.testInd = testInd;

net = init(net);

view(net);

net.trainParam.epochs = 600;

net.trainParam.max\_fail = 600;

net.trainParam.goal = 1.0e-8;

[Xs, Xi, Ai, Ts] = preparets(net, con2seq(x), {}, con2seq(y));

net = train(net, Xs, Ts, Xi, Ai);

Y = sim(net, Xs, Xi, Ai);

figure;

hold on;

grid on;

plot(tStart : h : tEnd, x, '-b', tStart : h : tEnd, [x(1:D) cell2mat(Y)], '-r')

legend('Source', 'Predicted');

figure;

hold on;

grid on;

plot(tStart+D\*h : h : tEnd, x(D+1:end) - cell2mat(Y), '-r')

legend('Train Error');

xValid = x( valInd( length(valInd) - (D - 1) : length(valInd) ) );

uValid = u( valInd( length(valInd) - (D - 1) : length(valInd) ) );

inpuT   = [con2seq( u(testInd)  ); con2seq( x(testInd) )];

delay = [con2seq( uValid ); con2seq(xValid)];

predictTest = sim(net, inpuT, delay, Ai);

figure;

hold on;

grid on;

plot(x(testInd), '-b')

plot(cell2mat(predictTest), '-r');

legend('Source', 'Predicted');

figure;

hold on;

grid on;

error = cell2mat(predictTest) - x(testInd);

plot(error, '-r');

legend('Test Error');

Выводы:

Такие сети используются в качестве ассоциативной памяти, а также в задачах нелинейной фильтрации данных, для предсказывания и распознавания закономерностей.

Научился применять динамические сети для предсказания значений временного ряда и выполнения многошагового прогноза, распознавания динамических образов, аппроксимации траектории динамической системы и выполнения многошагового прогноза.