**Лабораторная работа № 5**

«Сети с обратными связями Элмана, Хопфилда, Хэмминга ».

Постановка задачи.

Целью работы является исследование свойств сетей **Хопфилда**, **Хэмминга** и **Элмана**, алгоритмов обучения, а также применение сетей в задачах распознавания статических и динамических образов.

Ход работы.

1. Построить и обучить сеть **Элмана** для распознавания динамических образов. Проверить качество распознавания.
2. Построить и обучить сеть **Хопфилда** для распознавания статических образов. Проверить качество распознавания.
3. Построить и обучить сеть **Хэмминга** для распознавания статических образов. Проверить качество распознавания.

Теория.

Нейронная сеть **Элмана** — один из видов рекуррентных сетей, которая получается из многослойного персептрона введением обратных связей. Связи идут не от выхода сети, а от выходов внутреннего слоя. Это позволяет учесть предысторию наблюдаемых процессов и накопить информацию для выработки правильной стратегии управления. Данная нейронная сеть может применяться в системах управления движущимися объектами, так как её главной особенностью является запоминание последовательностей.

Нейронная сеть **Хопфилда** - полносвязная нейронная сеть с симметричной матрицей связей. В процессе работы динамика таких сетей сходится к одному из положений равновесия. Эти положения равновесия определяются заранее в процессе обучения. Они являются локальными минимумами функционала, который принято называть энергией сети.

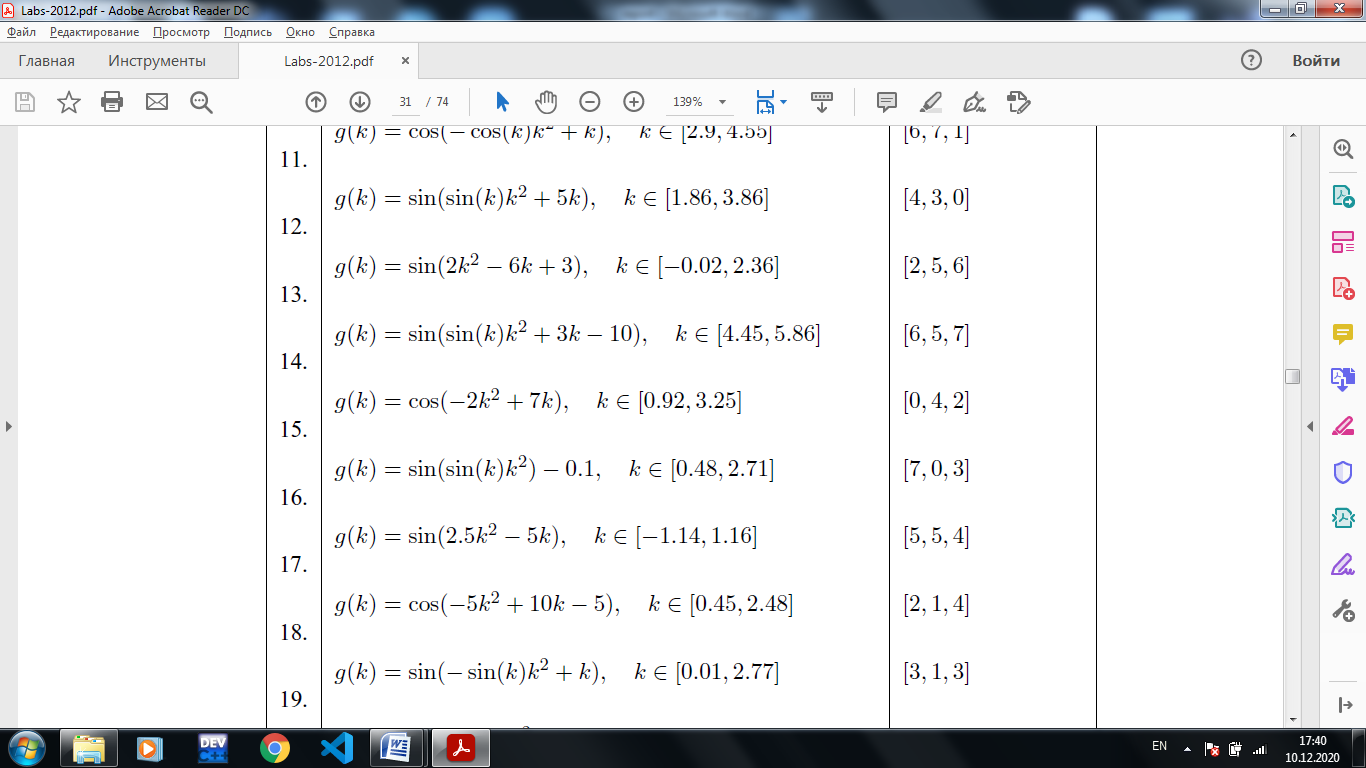
Такая сеть может быть использована как фильтр или как автоассоциативная память. В отличие от многих нейронных сетей, работающих до получения ответа через определённое количество тактов, сети Хопфилда работают до достижения равновесия, когда следующее состояние сети в точности равно предыдущему.

Нейронная сеть **Хэмминга** — вид нейронной сети, использующийся для классификации бинарных векторов, основным критерием в которой является расстояние Хэмминга. Является развитием нейронной сети Хопфилда.

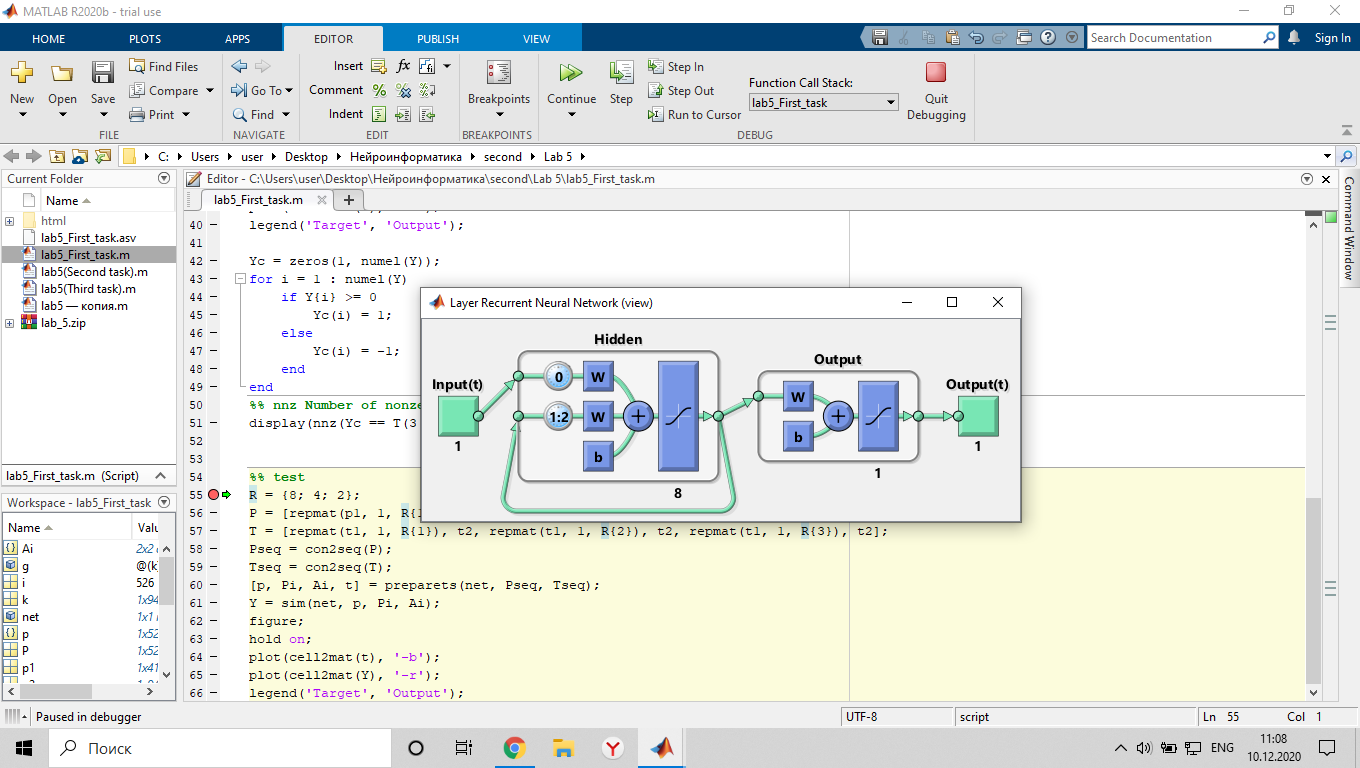
Стоит ещё раз отметить, что Нейронные сети **Хопфилда** и **Хэмминга** не работают по принципу «классического нейросетевого обучения». Из-за этого возникает серьезный недостаток, про который обязательно нужно упомянуть. Если сети подать на вход изображение, немного сдвинутое или слегка повернутое, то, скорее всего она не сможет нормально распознать изображение. Из плюсов стоит отметить, что эти сети справляются с хорошо зашумленными данными. Степень зашумленности в некоторых случаях может достигать 50%.

Входные данные и результаты.

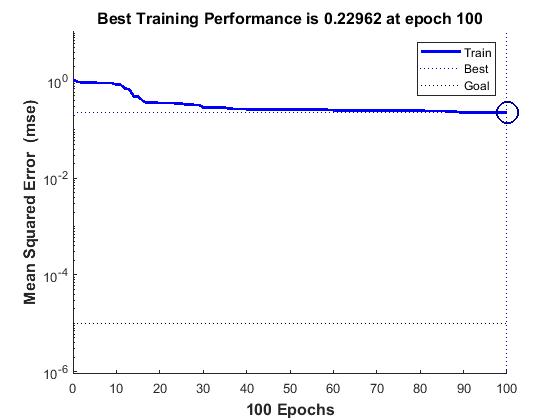
Вариант № 15.

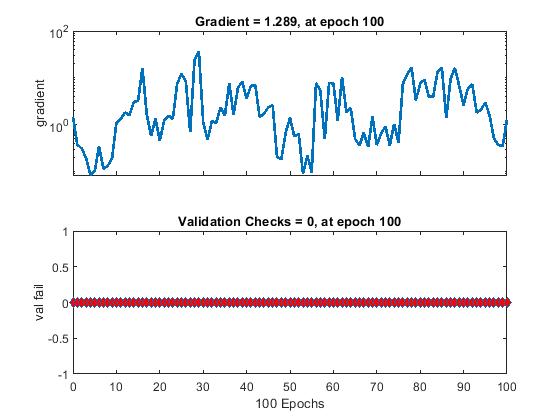


**Первое задание**

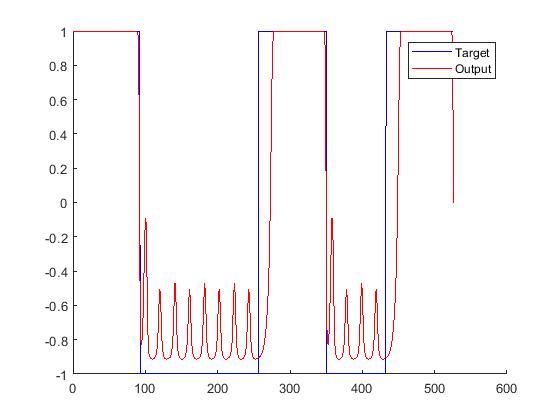


Performance



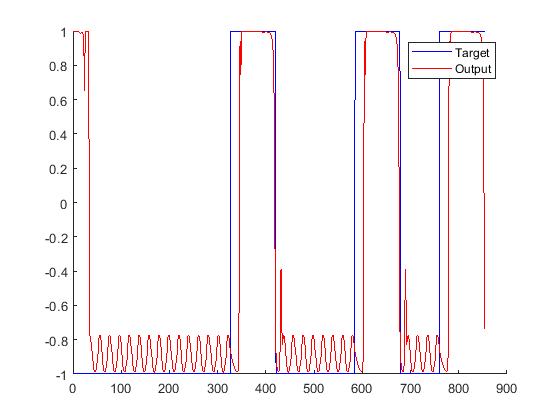


Train/Test

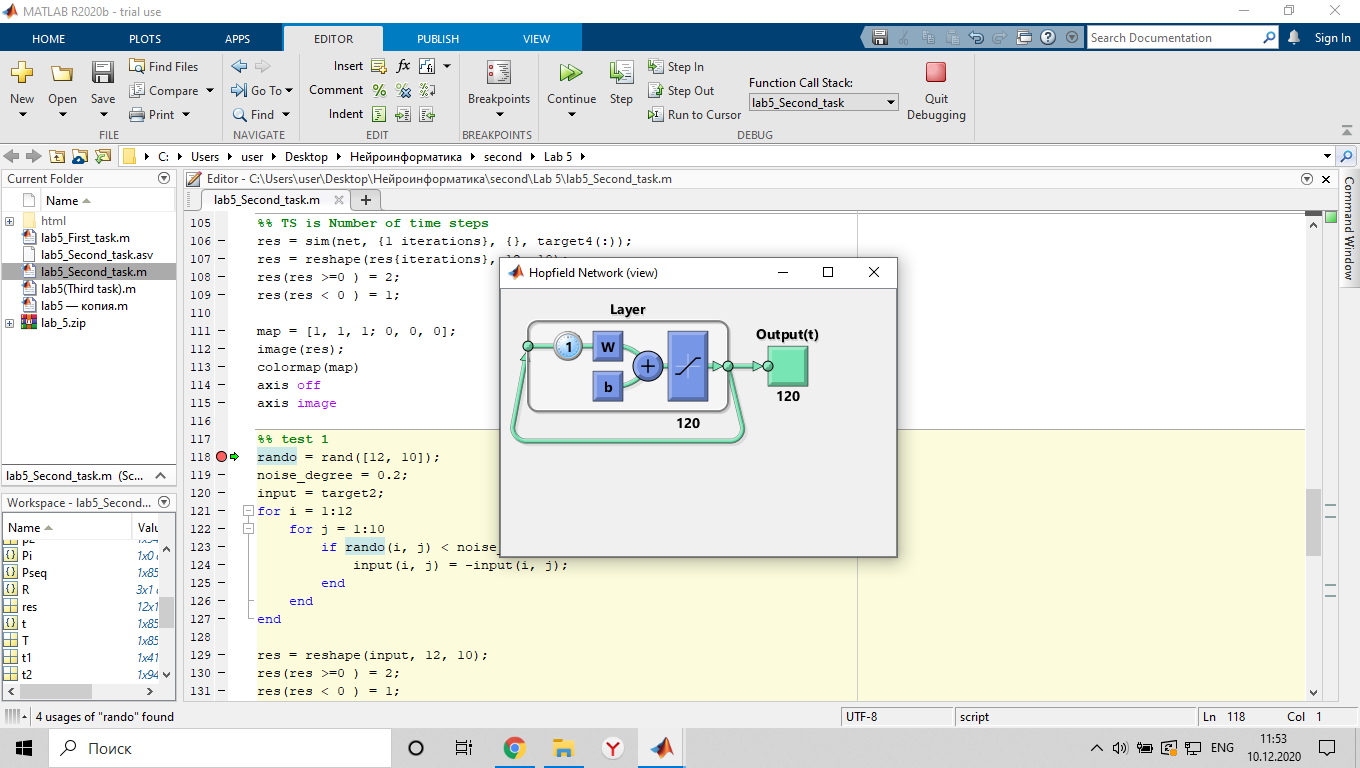


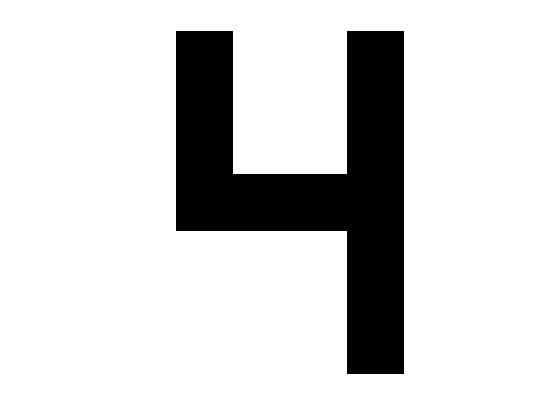
Правильно классифицированных точек: **737**

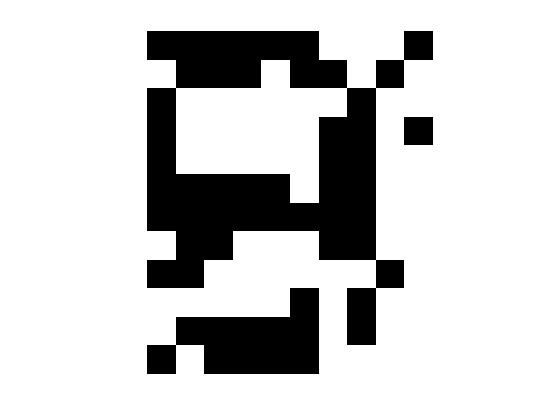
5 Random Points

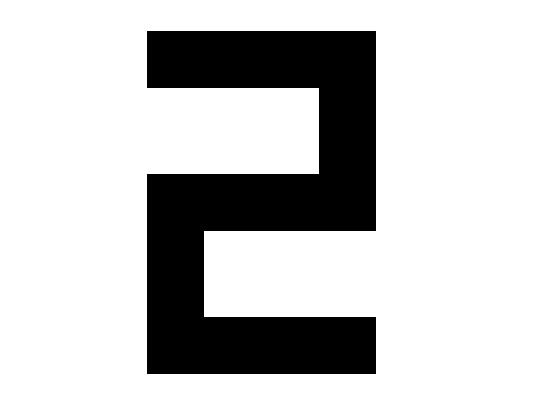


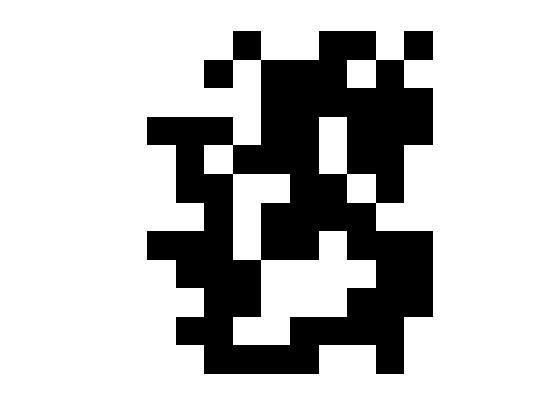
**Второе задание**

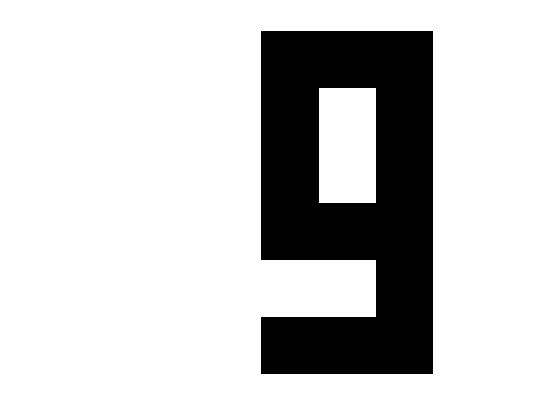
****



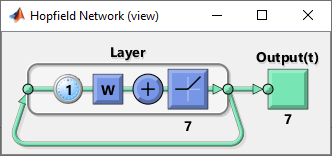


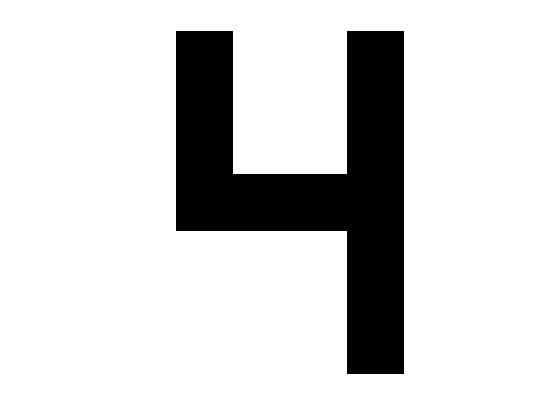


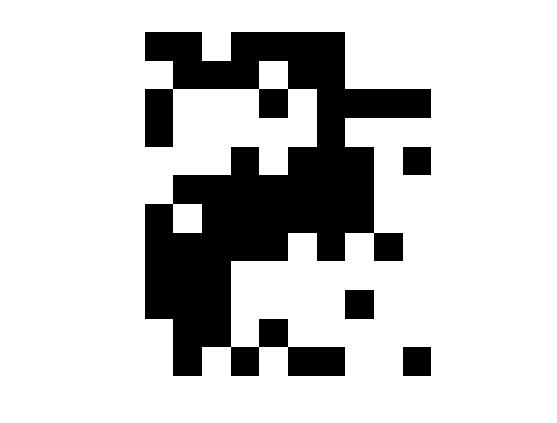


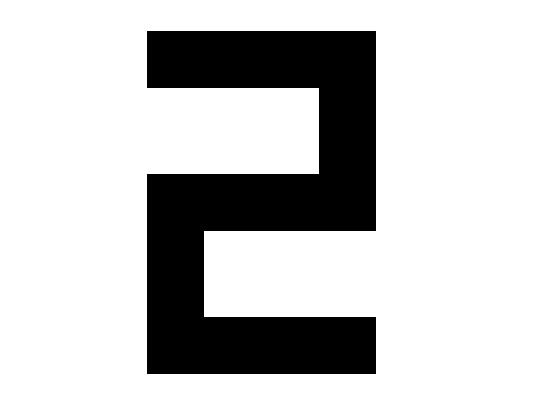


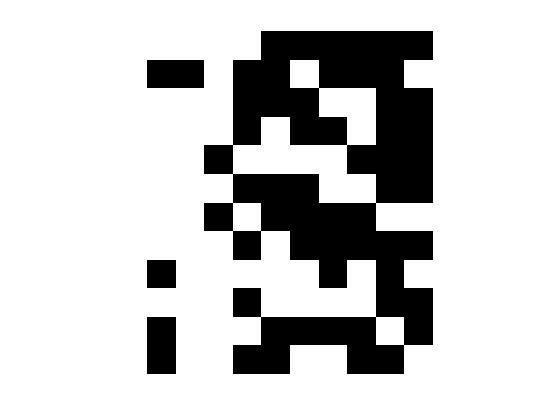
**Третье задание**

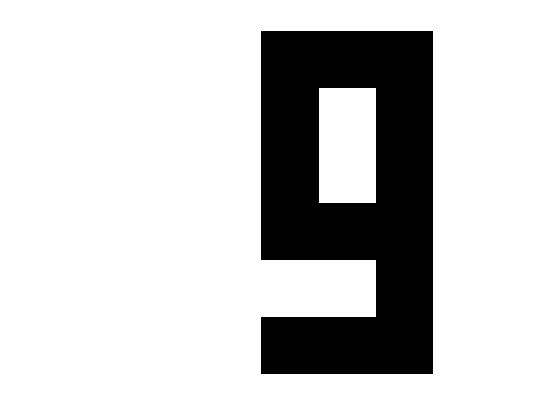












Исходный код.

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 = layrecnet(1 : 2, 8, 'trainoss');

net.layers{1}.transferFcn = 'tansig';

net.layers{2}.transferFcn = 'tansig';

net = configure(net, Pseq, Tseq);

view(net);

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

%% [Xs,Xi,Ai,Ts,EWs,shift] = preparets(net,Xnf,Tnf,Tf,EW)

[p, Pi, Ai, t] = preparets(net, Pseq, Tseq);

net.trainParam.epochs = 100;

net.trainParam.goal = 1.0e-5;

net = train(net, p, t, Pi, Ai);

Y = sim(net, p, Pi, Ai);

figure;

hold on;

plot(cell2mat(t), '-b');

plot(cell2mat(Y), '-r');

legend('Target', 'Output');

Yc = zeros(1, numel(Y));

for i = 1 : numel(Y)

    if Y{i} >= 0

        Yc(i) = 1;

    else

        Yc(i) = -1;

    end

end

%% nnz Number of nonzero matrix elements

display(nnz(Yc == T(3 : end)))

%% test

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);

[p, Pi, Ai, t] = preparets(net, Pseq, Tseq);

Y = sim(net, p, Pi, Ai);

figure;

hold on;

plot(cell2mat(t), '-b');

plot(cell2mat(Y), '-r');

legend('Target', 'Output');

clear;

clc;

target0 = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 +1 +1 +1 +1 +1 +1 -1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;

           -1 -1 +1 +1 +1 +1 +1 +1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 -1 -1 -1 -1 -1 -1 -1];

target1 = [-1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;

           -1 -1 -1 +1 +1 +1 +1 -1 -1 -1];

target2 = [+1 +1 +1 +1 +1 +1 +1 +1 -1 -1;

           +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;

           -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;

           -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;

           -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;

           +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;

           +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;

           +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;

           +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;

           +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;

           +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;];

target3 = [-1 -1 +1 +1 +1 +1 +1 +1 -1 -1;

           -1 -1 +1 +1 +1 +1 +1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 +1 +1 +1 +1 -1 -1;

           -1 -1 -1 -1 +1 +1 +1 +1 -1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 +1 +1 +1 +1 +1 +1 +1 -1;

           -1 -1 +1 +1 +1 +1 +1 +1 -1 -1];

target4 = [-1 +1 +1 -1 -1 -1 -1 +1 +1 -1;

           -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;

           -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;

           -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;

           -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;

           -1 +1 +1 +1 +1 +1 +1 +1 +1 -1;

           -1 +1 +1 +1 +1 +1 +1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;

           -1 -1 -1 -1 -1 -1 -1 +1 +1 -1];

target6 = [+1 +1 +1 +1 +1 +1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;

           +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;

           +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;

           +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;

           +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;

           +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;

           +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;

           +1 +1 +1 +1 +1 +1 -1 -1 -1 -1];

target9 = [-1 -1 -1 -1 +1 +1 +1 +1 +1 +1;

           -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;

           -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;

           -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;

           -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;

           -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;

           -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;

           -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;

           -1 -1 -1 -1 -1 -1 -1 -1 +1 +1;

           -1 -1 -1 -1 -1 -1 -1 -1 +1 +1;

           -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;

           -1 -1 -1 -1 +1 +1 +1 +1 +1 +1];

net = newhop([target4(:), target2(:), target9(:)]);

view(net);

iterations = 600;

%%[Y,Xf,Af] = sim(net,{Q TS},Xi,Ai)

%% Q is Batch size

%% TS is Number of time steps

res = sim(net, {1 iterations}, {}, target4(:));

res = reshape(res{iterations}, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

image(res);

colormap(map)

axis off

axis image

%% test 1

randomValues = rand([12, 10]);

noise\_degree = 0.2;

input = target2;

for i = 1:12

    for j = 1:10

        if randomValues(i, j) < noise\_degree

            input(i, j) = -input(i, j);

        end

    end

end

res = reshape(input, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Noised 0.2');

image(res);

colormap(map)

axis off

axis image

res = sim(net, {1 iterations}, {}, input(:));

res = reshape(res{iterations}, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Recognised 0.2');

image(res);

colormap(map)

axis off

axis image

%% test 2

randomValues = rand([12, 10]);

noise\_degree = 0.3;

input = target9;

for i = 1:12

    for j = 1:10

        if randomValues(i, j) < noise\_degree

            input(i, j) = -input(i, j);

        end

    end

end

res = reshape(input, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Noised 0.3');

image(res);

colormap(map)

axis off

axis image

res = sim(net, {1 iterations}, {}, input(:));

res = reshape(res{iterations}, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Recognised 0.3');

image(res);

colormap(map)

axis off

axis image

p = [target0(:), target1(:), target2(:), target3(:), target4(:), target6(:), target9(:)];

Q = 7;      %% Число эталонных образцов

R = 120;    %% Размерность входного вектора

eps = 1 / (Q - 1);

IW = [target0(:)';

      target1(:)';

      target2(:)';

      target3(:)';

      target4(:)';

      target6(:)';

      target9(:)'];

b = ones(Q, 1) \* R;

a = zeros(Q, Q);

for i = 1 : Q

    a(:, i) = IW \* p(:, i) + b;

end

net = newhop(a);

net.biasConnect(1) = 0;

net.layers{1}.transferFcn = 'poslin';

net.LW{1, 1} = eye(Q, Q) \* (1 + eps) - ones(Q, Q) \* eps;

view(net);

iterations = 600;

input = target4(:);

a1 = IW \* input + b;

res = sim(net, {1 iterations}, {}, a1);

a2 = res{iterations};

ind = find(a2 == max(a2));

answer = IW(ind, :)';

res = reshape(answer, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

image(res);

colormap(map)

axis off

axis image

%% test 1

input = target2;

rando = rand([12, 10]);

noise\_degree = 0.3;

for i = 1:12

    for j = 1:10

        if rando(i, j) < noise\_degree

            input(i, j) = -input(i, j);

        end

    end

end

res = reshape(input, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Noised');

image(res);

colormap(map)

axis off

axis image

input = input(:);

a1 = IW \* input + b;

res = sim(net, {1 iterations}, {}, a1);

a2 = res{iterations};

ind = find(a2 == max(a2));

answer = IW(ind, :)';

figure

res = reshape(answer, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

image(res);

colormap(map)

axis off

axis image

%% test 2

input = target9;

rando = rand([12, 10]);

noise\_degree = 0.3;

for i = 1:12

    for j = 1:10

        if rando(i, j) < noise\_degree

            input(i, j) = -input(i, j);

        end

    end

end

res = reshape(input, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

figure('Name', 'Noised');

image(res);

colormap(map)

axis off

axis image

input = input(:);

a1 = IW \* input + b;

res = sim(net, {1 iterations}, {}, a1);

a2 = res{iterations};

ind = find(a2 == max(a2));

answer = IW(ind, :)';

figure

res = reshape(answer, 12, 10);

res(res >=0 ) = 2;

res(res < 0 ) = 1;

map = [1, 1, 1; 0, 0, 0];

image(res);

colormap(map)

axis off

axis image

Выводы:

Научился использовать сети **Хопфилда**, **Хэмминга** для распознавания статических образов и сеть **Элмана** для распознавания динамических образов.

Понял, что у нейронной сети **Хопфилда** есть ряд недостатков.

* Во-первых, это относительно небольшой объём памяти, величину которой можно оценить как 15% от размера.
* Во-вторых, достижение устойчивого состояния не гарантирует правильный ответ сети. Это происходит из-за того, что сеть может сойтись к так называемым ложным аттракторам, иногда называемыми «химерами».