МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)
Кафедра вычислительной математики и программирования

Лабораторная работа №5
по спецкурсу «Нейроинформатика»

Сети с обратными связями

Выполнил: Днепров И.С.
Группа: М8О-407Б, вариант 10
Преподаватели: Тюменцев Ю.В.

Москва, 2020

Цель работы

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

Основные этапы работы:

1. Использовать сеть Элмана для распознавания динамических образов. Проверить качество распознавания.

2. Использовать сеть Хопфилда для распознавания статических образов. Проверить качество распознавания.

3. Использовать сеть Хэмминга для распознавания статических образов. Проверить качество распознавания.

Оборудование

Процессор: 2,4 GHz Intel Core 2 Duo

Оперативная память: 8 ГБ 1067 MHz DDR3

Программное обеспечение

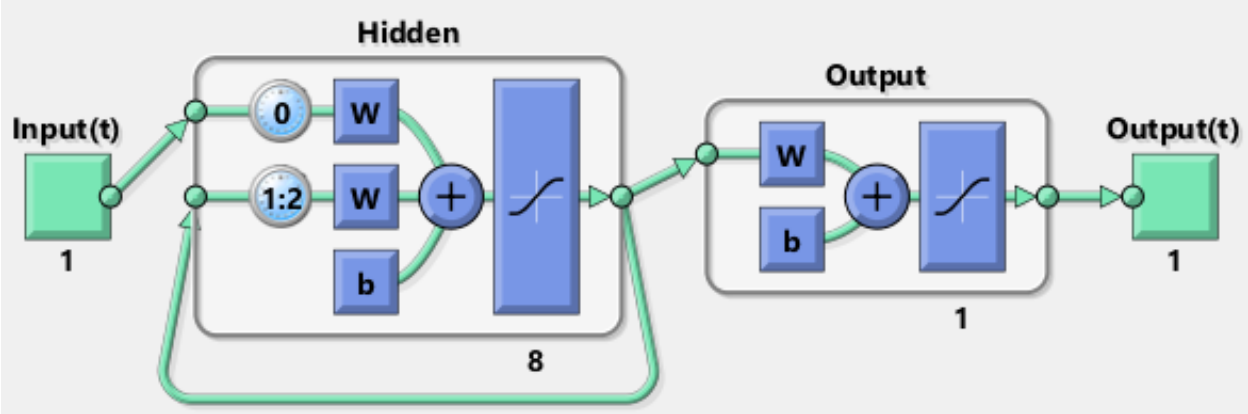
Matlab R2020b, 64-bit.

Задание 1.

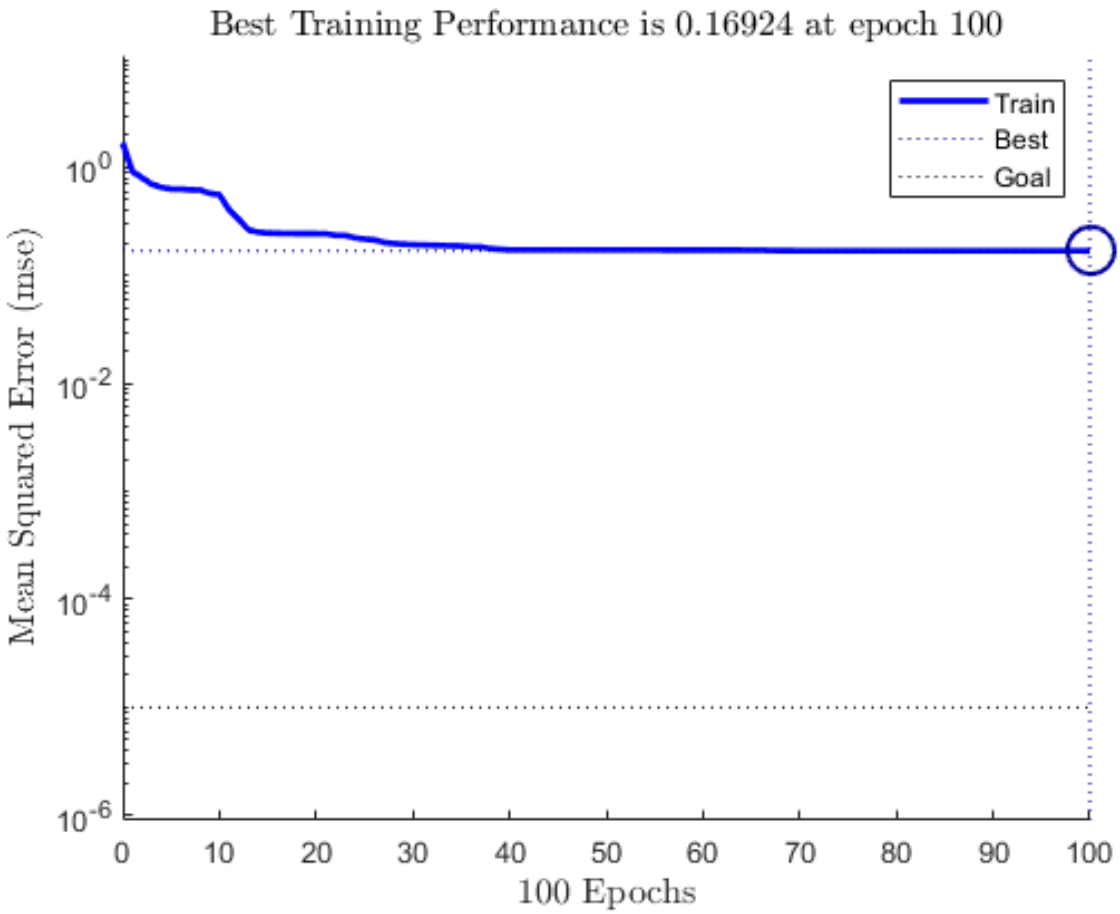
Вариант задания:

$$g(k) = \sin\left(-3k^2 + 5k + 10\right) + 0.8, \quad k \in [0.46,3.01], \quad R = [0,2,2]$$

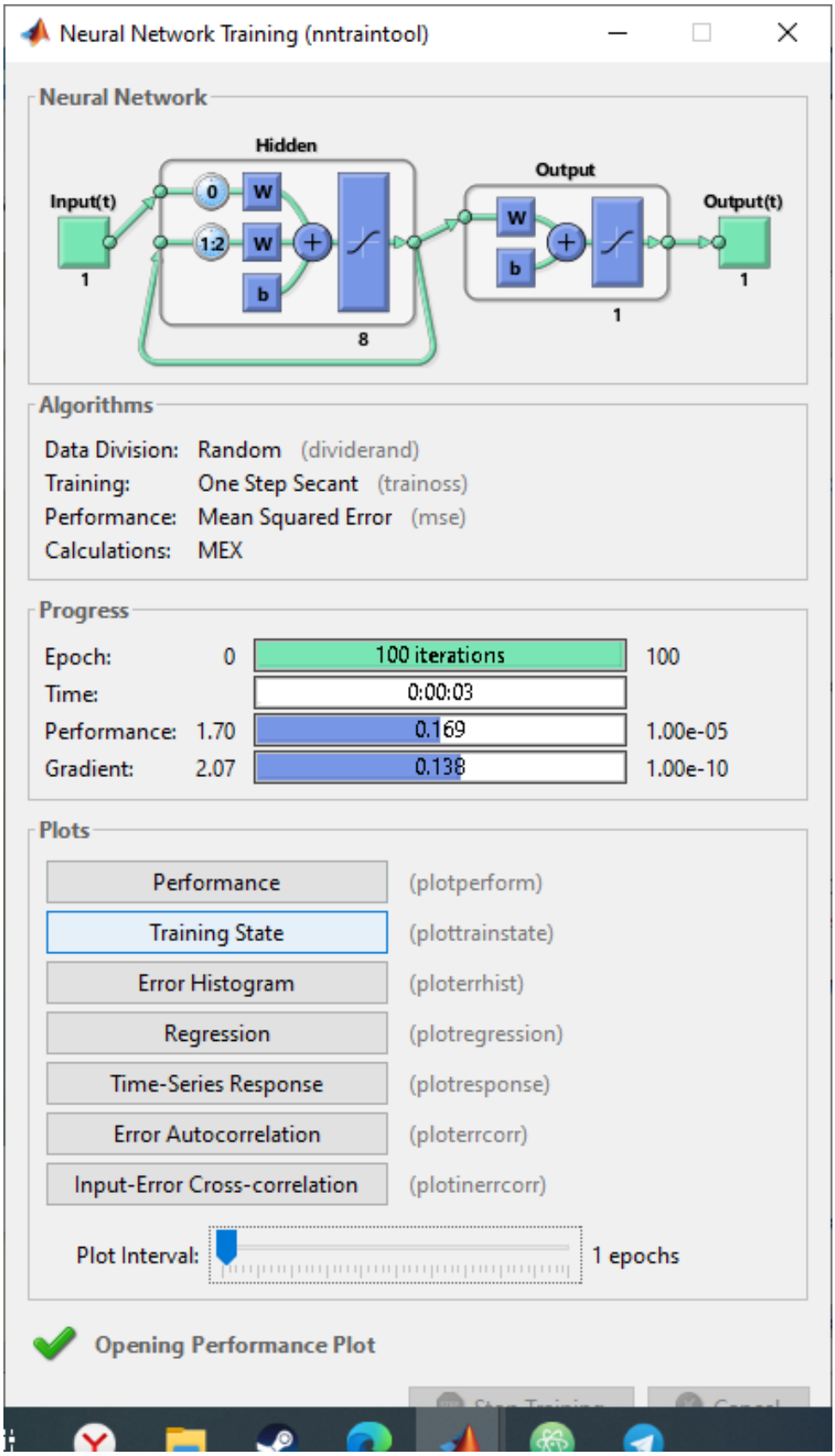
Структура сети:



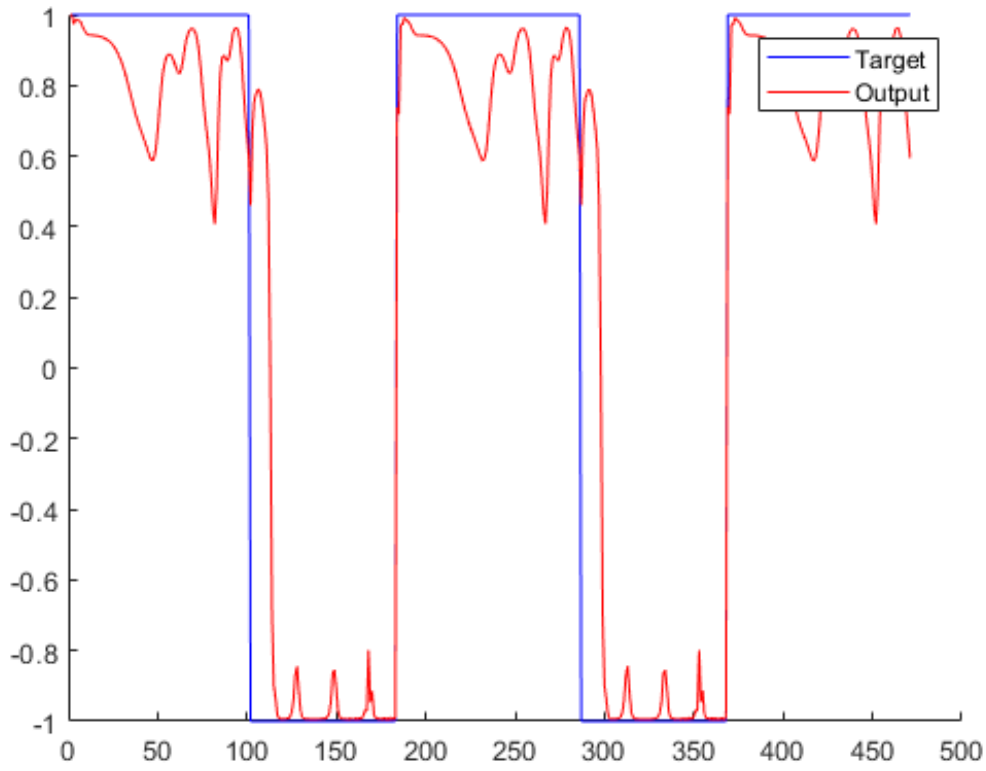
Сходимость ошибки:



Обучение сети:

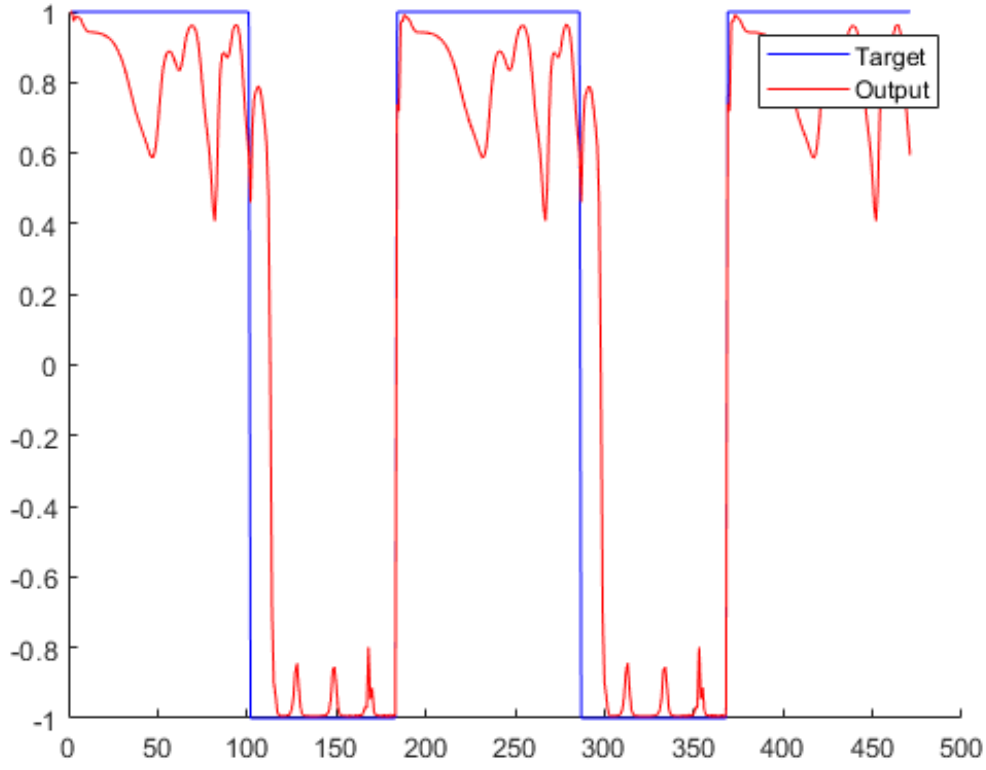


Результаты на обучающей выборке:



Результаты на тестовой выборке:

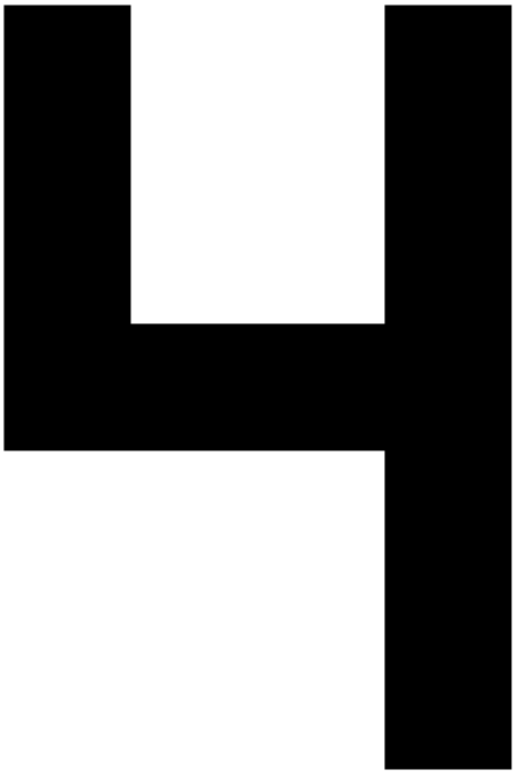
Right classified 449\471



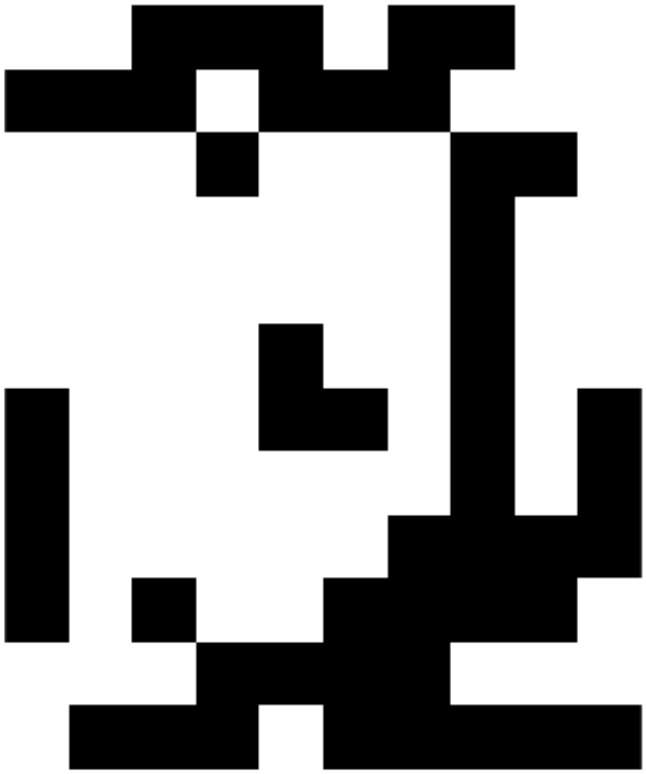
Задание 2.
Вариант задания:

[4,3,0]

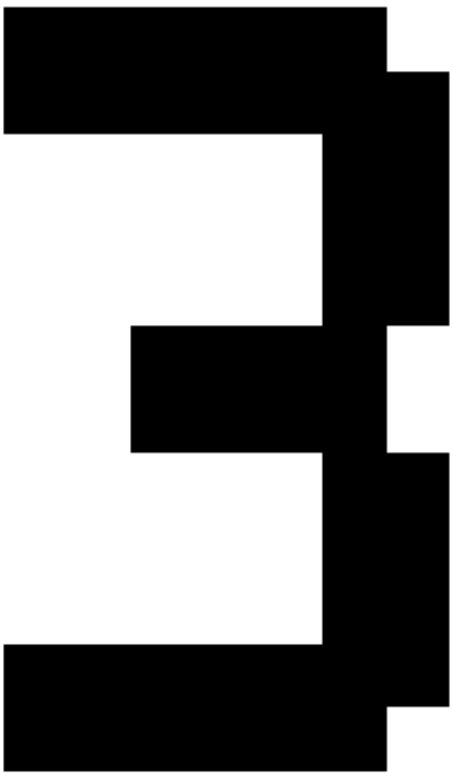
Результат распознавания первого образца:



Второй образец после зашумления на 20%:



Результат распознавания второго образца после зашумления:



Третий образец после зашумления на 30%:

Результат распознавания третьего образца после зашумления:

Код программы

Lab4.m

```
set(0, 'DefaultTextInterpreter', 'latex');
% Задание 1
% Задаем динамический образ
k = 0 : 0.025 : 1;
p1 = sin(4 * pi * k);
t1 = -ones(size(k));

k = 0.46 : 0.025 : 3.01;
g = @(k) cos( -3 * k .^ 2 + 5 * k + 10) + 0.8;
p2 = g(k);
t2 = ones(size(k));

% Задаем длительность и обучающее множество
R = {0; 2; 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];
PLearn = con2seq(P);
TLearn = con2seq(T);

% Создаем сеть с задержками от 1 до 2 и 8 нейронами на скрытом слое с обучающей
функцией tansig
network = layrecnet(1 : 2, 8, 'trainoss');
network.layers{1}.transferFcn = 'tansig';
network.layers{2}.transferFcn = 'tansig';
network = configure(network, PLearn, TLearn);
view(network);

% Готовим данные для обучения
[p, Pi, Ai, t] = preparets(network, PLearn, TLearn);

%% Обучаем сеть
network.trainParam.epochs = 100;
network.trainParam.goal = 1.0e-5;
network = train(network, p, t, Pi, Ai);
Y = sim(network, p, Pi, Ai);

%% Результаты обучения
figure;
hold on;
plot(cell2mat(t), '-b');
plot(cell2mat(Y), '-r');
legend('Target', 'Output');

% Преобразуем значения
Yc = zeros(1, numel(Y));
for i = 1 : numel(Y)
    if Y{i} >= 0
        Yc(i) = 1;
    else
        Yc(i) = -1;
```

```
    end
end
fprintf('\nRight classified %d\\%d\n', nnz(Yc == T(3 : end)), numel(Y));

%% Проверка качества обучения
R = {0; 2; 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];
PLearn = con2seq(P);
TLearn = con2seq(T);
[p, Pi, Ai, t] = preparets(network, PLearn, TLearn);
Y = sim(network, p, Pi, Ai);
figure;

hold on;
plot(cell2mat(t), '-b');
plot(cell2mat(Y), '-r');
legend('Target', 'Output');

%% Задание 2
% Задаем цифры попиксельно
p0 = [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 +1 +1 +1 +1 +1 +1 -1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 +1 +1 +1 -1 -1 +1 +1 +1 -1;
      -1 -1 +1 +1 +1 +1 +1 +1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 -1 -1 -1 -1 -1 -1 -1];

p1 = [-1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1;
      -1 -1 -1 +1 +1 +1 +1 -1 -1 -1];

p2 = [+1 +1 +1 +1 +1 +1 +1 +1 -1 -1;
      +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;
      -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;
      -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;
      -1 -1 -1 -1 -1 -1 +1 +1 -1 -1;
      +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;
      +1 +1 +1 +1 +1 +1 +1 +1 -1 -1;
      +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;
```

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

p3 = [-1 -1 +1 +1 +1 +1 +1 +1 -1 -1;
      -1 -1 +1 +1 +1 +1 +1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 +1 +1 +1 +1 -1 -1;
      -1 -1 -1 -1 +1 +1 +1 +1 -1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 +1 +1 +1 +1 +1 +1 +1 -1;
      -1 -1 +1 +1 +1 +1 +1 +1 -1 -1];

p4 = [-1 +1 +1 -1 -1 -1 -1 +1 +1 -1;
      -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;
      -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;
      -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;
      -1 +1 +1 -1 -1 -1 -1 +1 +1 -1;
      -1 +1 +1 +1 +1 +1 +1 +1 +1 -1;
      -1 +1 +1 +1 +1 +1 +1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1;
      -1 -1 -1 -1 -1 -1 -1 +1 +1 -1];

p6 = [+1 +1 +1 +1 +1 +1 -1 -1 -1 -1;
      +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;
      +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;
      +1 +1 -1 -1 -1 -1 -1 -1 -1 -1;
      +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;
      +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;
      +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;
      +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;
      +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;
      +1 +1 -1 -1 +1 +1 -1 -1 -1 -1;
      +1 +1 +1 +1 +1 +1 -1 -1 -1 -1;
      +1 +1 +1 +1 +1 +1 -1 -1 -1 -1];

p9 = [-1 -1 -1 -1 +1 +1 +1 +1 +1 +1;
      -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;
      -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;
      -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;
      -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;
      -1 -1 -1 -1 +1 +1 -1 -1 +1 +1;
      -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;
      -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;
      -1 -1 -1 -1 -1 -1 -1 -1 +1 +1;
      -1 -1 -1 -1 -1 -1 -1 -1 +1 +1;
      -1 -1 -1 -1 +1 +1 +1 +1 +1 +1;
```

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

%% Создаем сеть и обучаем ее
network = newhop([p4(:), p3(:), p0(:)]);
iterations = 600;
resultImg = sim(network, {1 iterations}, {}, p4(:));

% Преобразуем изображение к нужному виду
resultImg = reshape(resultImg{iterations}, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Результаты
map = [1, 1, 1; 0, 0, 0];
figure('Name', '4');
image(resultImg);
colormap(map)
axis off
axis image

%% Проверка работы сети при зашумлении 0.2
randomPixels = rand([12, 10]);
M = 0.2;
img = p3;

% Зашумление изображения
for i = 1:12
    for j = 1:10
        if randomPixels(i, j) < M
            img(i, j) = -img(i, j);
        end
    end
end

% Преобразуем изображение к нужному виду
resultImg = reshape(img, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', '3 noise 20%');
image(resultImg);
colormap(map)
axis off
axis image

%% Распознаем зашумленное изображение
resultImg = sim(network, {1 iterations}, {}, img(:));

% Преобразуем изображение к нужному виду
resultImg = reshape(resultImg{iterations}, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;
```

```
% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', 'network should return 3');
image(resultImg);
colormap(map)
axis off
axis image

%% Проверка работы сети при зашумлении 0.3
randomPixels = rand([12, 10]);
M = 0.3;
img = p0;

% Зашумление изображения
for i = 1:12
    for j = 1:10
        if randomPixels(i, j) < M
            img(i, j) = -img(i, j);
        end
    end
end

% Преобразуем изображение к нужному виду
resultImg = reshape(img, 12, 10);
resultImg(resultImg >=0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', '0 noise 30%');
image(resultImg);
colormap(map)
axis off
axis image

%% Распознаем зашумленное изображение
resultImg = sim(network, {1 iterations}, {}, img(:));

% Преобразуем изображение к нужному виду
resultImg = reshape(resultImg{iterations}, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', 'network should return 0');
image(resultImg);
colormap(map)
axis off
axis image

%% Задание 3
% Готовим данные для обучения
```

```
p = [p0(:), p1(:), p2(:), p3(:), p4(:), p6(:), p9(:)];
IW = [p0(:)'; p1(:)'; p2(:)'; p3(:)'; p4(:)'; p6(:)'; p9(:)'];
R =  numel(p0);
Q = numel(IW) / R;
eps = 1 / (Q - 1);
b = ones(Q, 1) * R;
a = zeros(Q, Q);
for i = 1 : Q
    a(:, i) = IW * p(:, i) + b;
end

% Создаем нейросеть
network = newhop(a);
network.biasConnect(1) = 0;
network.layers{1}.transferFcn = 'poslin';
network.LW{1, 1} = eye(Q, Q) * (1 + eps) - ones(Q, Q) * eps;

% Обучем сеть
iterations = 600;
img = p4(:);
a1 = IW * img + b;
resultImg = sim(network, {1 iterations}, {}, a1);
a2 = resultImg{iterations};
idx = (a2 == max(a2));
answer = IW(idx, :)';

% Преобразуем изображение к нужному виду
resultImg = reshape(answer, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', '4');
image(resultImg);
colormap(map)
axis off
axis image

%% Проверка работы сети при зашумлении 0.2
img = p3;
randomPixels = rand([12, 10]);
M = 0.3;

% Зашумление изображения
for i = 1:12
    for j = 1:10
        if randomPixels(i, j) < M
       img(i, j) = -img(i, j);
        end
    end
end

% Преобразуем изображение к нужному виду
```

```
resultImg = reshape(img, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', '3 noise 20%');
image(resultImg);
colormap(map)
axis off
axis image

%% Распознаем зашумленное изображение
img = img(:);
a1 = IW * img + b;
resultImg = sim(network, {1 iterations}, {}, a1);
a2 = resultImg{iterations};
idx = (a2 == max(a2));
answer = IW(idx, :)';

% Преобразуем изображение к нужному виду
resultImg = reshape(answer, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', 'network should return 3');
image(resultImg);
colormap(map)
axis off
axis image

%% Проверка работы сети при зашумлении 0.3
img = p0;
randomPixels = rand([12, 10]);
M = 0.3;

% Зашумление изображения
for i = 1:12
    for j = 1:10
        if randomPixels(i, j) < M
       img(i, j) = -img(i, j);
        end
    end
end

% Преобразуем изображение к нужному виду
resultImg = reshape(img, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
```

```
figure('Name', '0 noise 30%');
image(resultImg);
colormap(map)
axis off
axis image


%% Распознаем зашумленное изображение
img = img(:);
a1 = IW * img + b;
resultImg = sim(network, {1 iterations}, {}, a1);
a2 = resultImg{iterations};
idx = (a2 == max(a2));
answer = IW(idx, :)';

% Преобразуем изображение к нужному виду
resultImg = reshape(answer, 12, 10);
resultImg(resultImg >= 0 ) = 2;
resultImg(resultImg < 0 ) = 1;

% Выводим результат
map = [1, 1, 1; 0, 0, 0];
figure('Name', 'network should return 0');
image(resultImg);
colormap(map)
axis off
axis image
```

Вывод

Данная работа показалась мне довольно интересной, особенно задания 2 и 3 с распознаванием цифр. Оказалось, что уже при 20% зашумленио я сам не способен распознать цифру, а нейросеть справляется с этой задачей без особых сложностей. Сеть Хопфилда, сколько вы раз я её не обучал и сколько эпох не выставлял, не смогла достичь того состояния, в котором все точки из выборки были классифицированы верно. Это обусловлено двумя существенными недостатками данной сети: небольшой объем памяти ($\frac{n}{2 \cdot \ln(n)}$) и отсутствие гарантий точности обучения (сеть может сойтись к ложным аттракторам).