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

Лабораторная работа №5
по спецкурсу «Нейроинформатика»

Сети с обратными связями

Выполнил: Дюсекеев А.Е.
Группа: М8О-101М-21
Преподаватель: Леонов С.С.

Москва, 2022

Цель работы

Исследование свойств сетей Хопфилда, Хэмминга и Элмана, алгоритмов 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

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

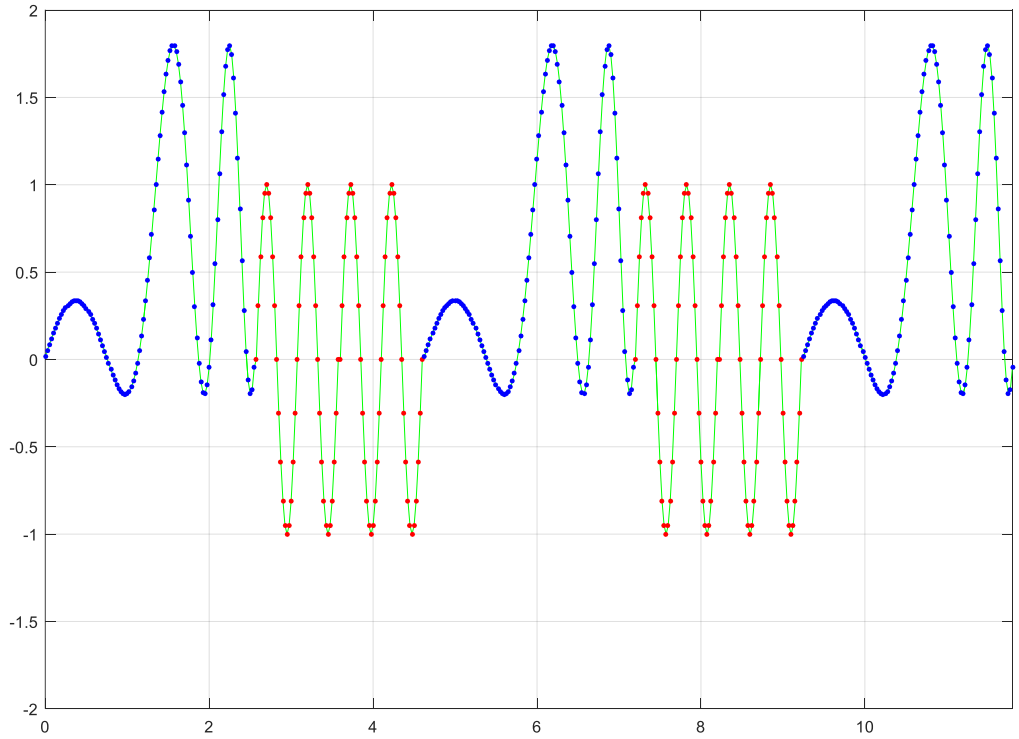
1. Построить и обучить сеть Элмана, которая будет выполнять распознавание динамического образа. Проверить качество распознавания.
   1.1. Входная последовательность обучающего множества состоит из комбинации основного сигнала (p_1) и сигнала, подлежащего распознаванию (p_2). Каждому значению основного сигнала соответствует -1 целевого выхода, каждому значению сигнала p_2 соответствует 1 целевого выхода.
      - $p_1(k) = \sin 4\pi k\,,\ t_1(k) = -1,\ k \in [0,1]$ с шагом $h = 0.025$
      - $p_2(k) = \sin(-3k^2 + 5k + 10) + 0.8\,, t_2(k) = 1, k \in [0.46, 3.01]$ с шагом $h = 0.025$

   Длительность основного сигнала задаётся набором чисел $R = \{0,2,2\}$.

   Входное множество определяется по формуле

```
X = [repmat(p1, 1, r1), p2, repmat(p1, 1, r2), p2, repmat(p1, 1, r3), p2];
y = [repmat(t1, 1, r1), t2, repmat(t1, 1, r2), t2, repmat(t1, 1, r3), t2];
```

Преобразовать обучающее множество с помощью функции *con2seq*. Не выделять из обучающего множества контрольное и тестовое подмножества.



1.2. Создать сеть с помощью функции *layrecnet*. Задать задержки $1:2$. Число нейронов скрытого слоя задать равным 8. Для обучения сети

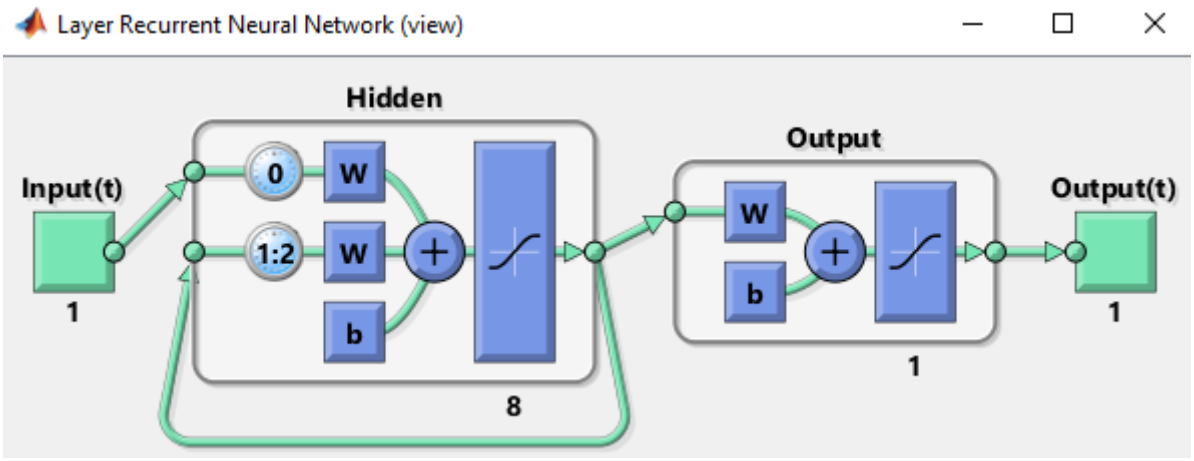
использовать одношаговый метод секущих (*trainoss*). Для скрытого и выходного слоев использовать *tansig* в качестве активационной функции (*net.layers{i}.transferFcn*). Сконфигурировать сеть (*configure*) под обучающее множество.

1.3. С помощью функции *preparets* сформировать массивы ячеек для функции обучения, содержащие обучающее множество и значения для инициализации задержек обратной связи (*P, T, Pi, Ai* соответственно). Если при выполнении заданий используется версия *MATLAB*, которая не поддерживает эту функцию, то обучать и выполнять расчет выходов сети без инициализации задержек.

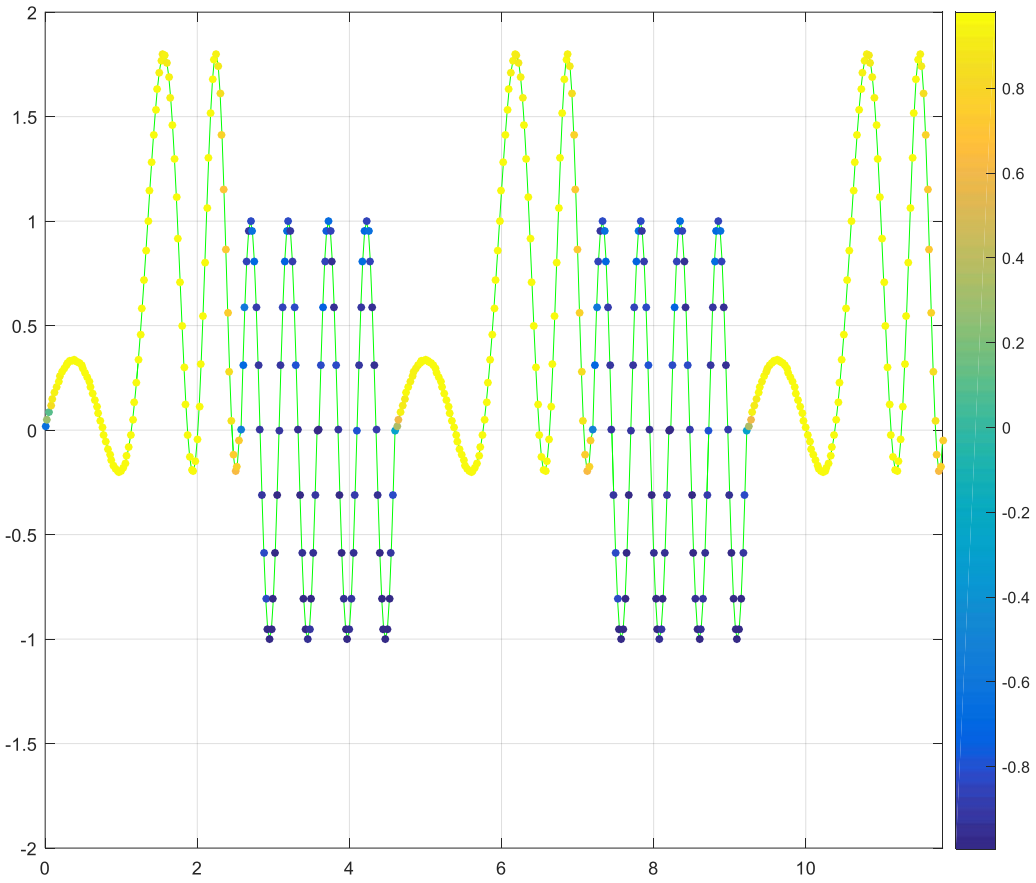
1.4. Задать параметры обучения: число эпох обучения (*net.trainParam.epochs*) равным 100, предельное значение критерия обучения (*net.trainParam.goal*) равным 10^{-5}.

1.5. Произвести обучение сети. Если необходимо, то произвести обучение несколько раз. Если результаты неудовлетворительные, то увеличить число нейронов сети. Занести в отчет содержимое *Performance* и *Neural Network Training*.

1.6.Отразить структуру сети и проведенное обучение



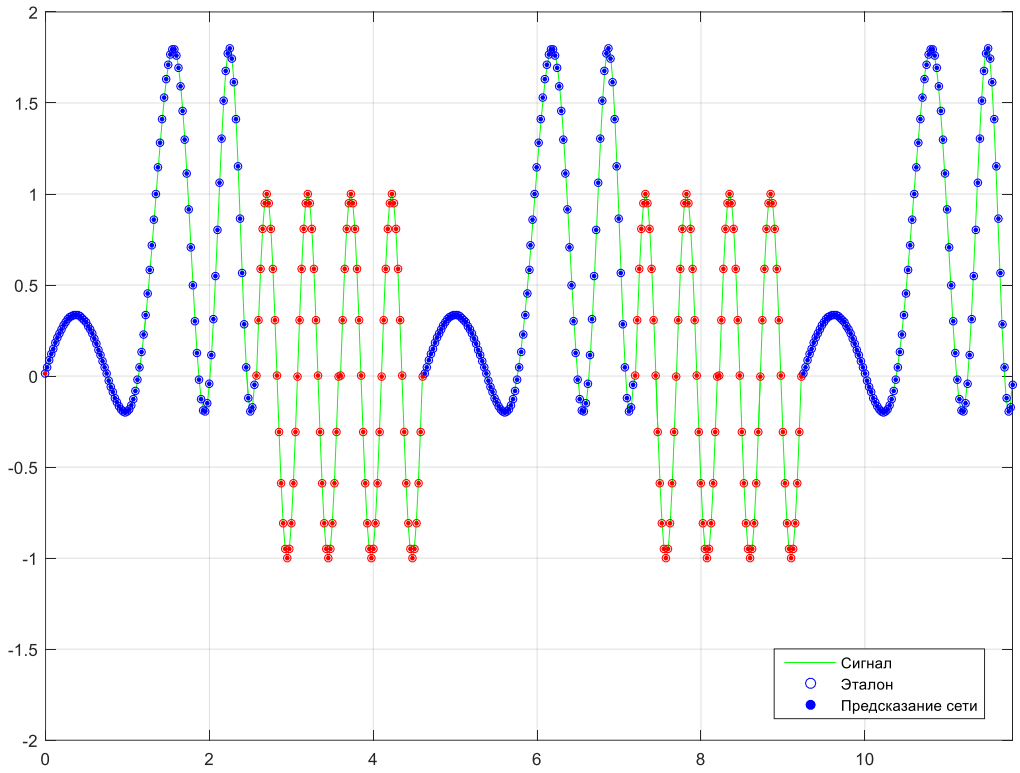
1.7. Рассчитать выход сети (*sim*) для обучающего подмножества. Отобразить на графике эталонные значения и предсказанные сетью. С помощью функции *legend* подписать кривые.



1.8. Преобразовать значения по правилу

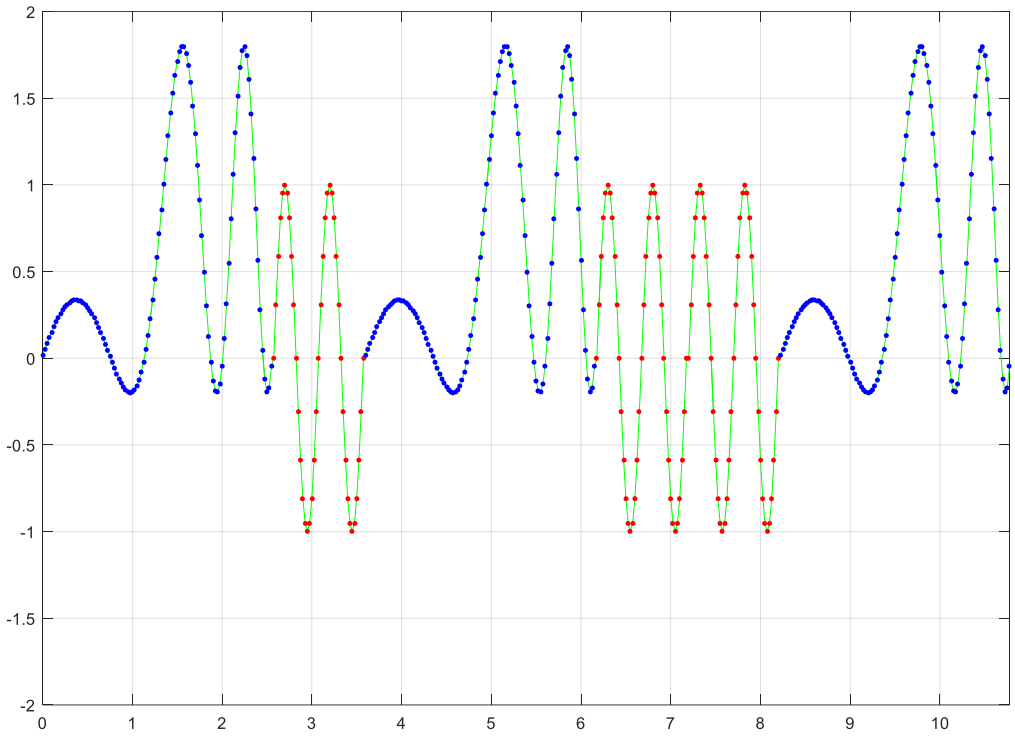
$$o_{ij} = \begin{cases} 1, & a_{ij} \geq 0 \\ -1, & a_{ij} < 0 \end{cases}$$

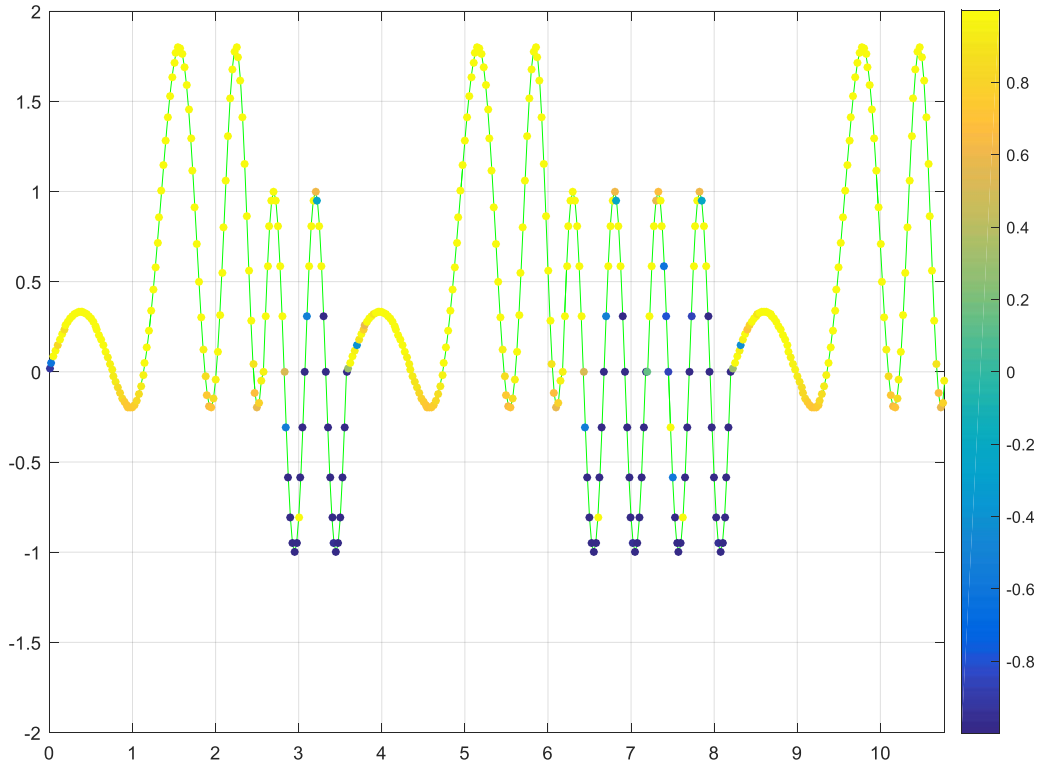
Сравнить выход сети с эталонными значениями. Занести в отчет количество правильно классифицированных точек.



Правильно классифицировано точек 472/473.

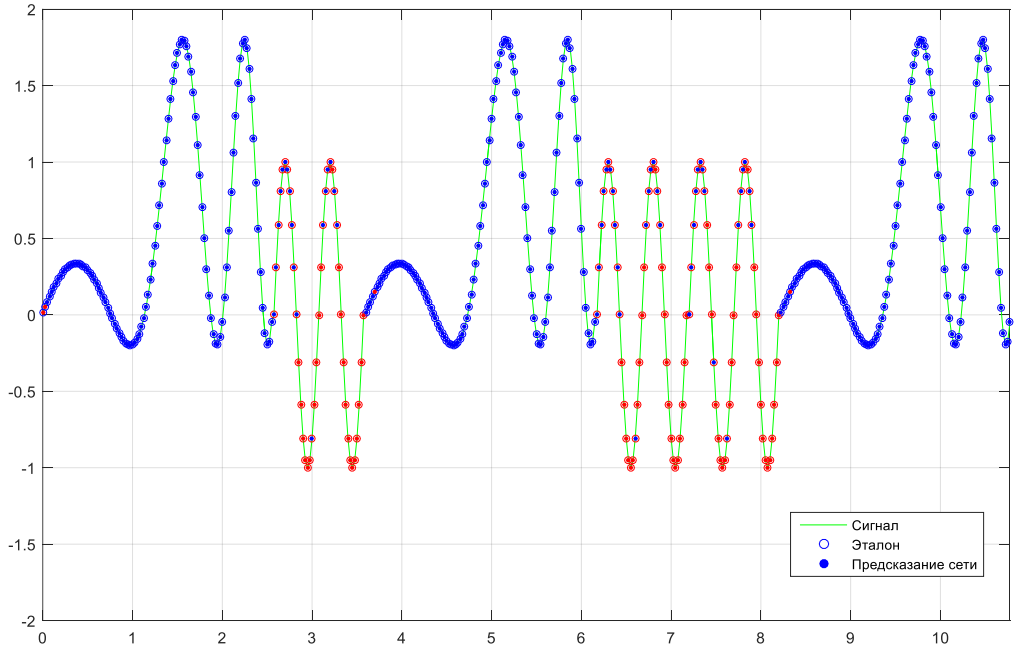
1.9. Для проверки качества распознавания сформировать новое обучающее множество, изменив одно из значений $R = \{0,2,2\}$. Рассчитать выходы сети для изменённой входной последовательности.


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
1
2
3
4
5
6
7
8
9
10
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8

1.10. Рассчитать выход сети (*sim*) для обучающего подмножества. Отобразить на графике эталонные значения и предсказанные сетью. С помощью функции *legend* подписать кривые.

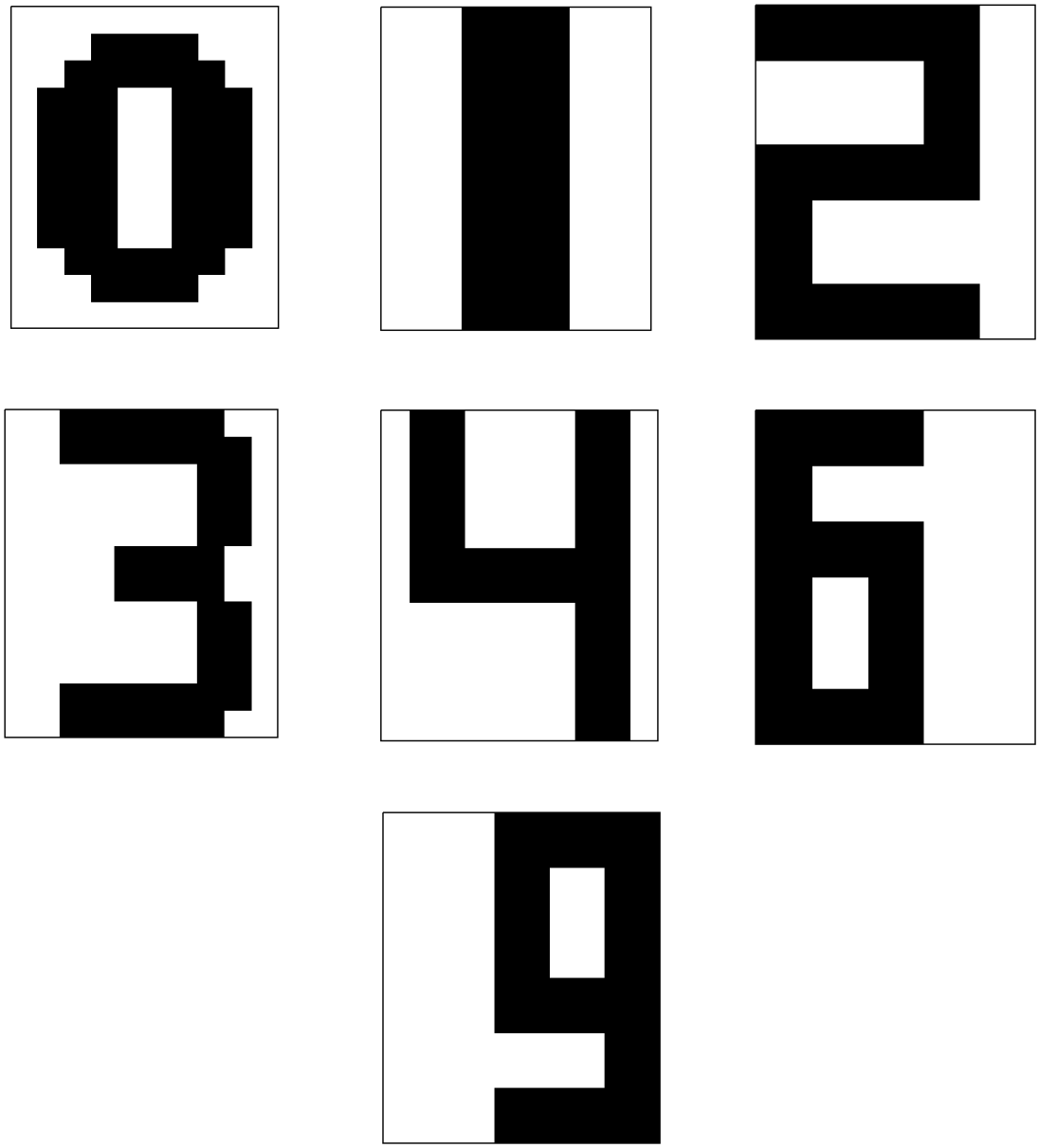
1.11. Преобразовать значения по правилу. Сравнить выход сети с эталонными значениями. Занести в отчёт количество правильно классфицированных точек.



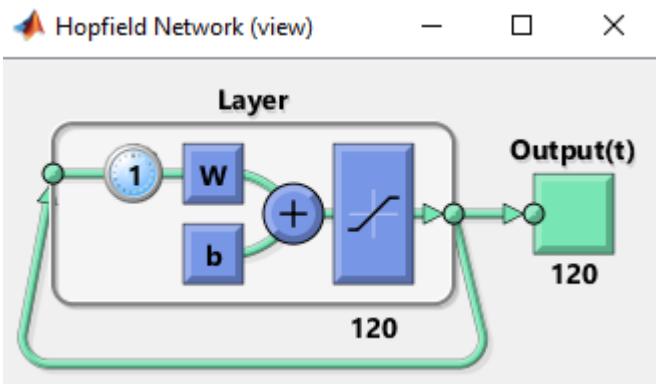
Правильно классифицировано точек 376/432

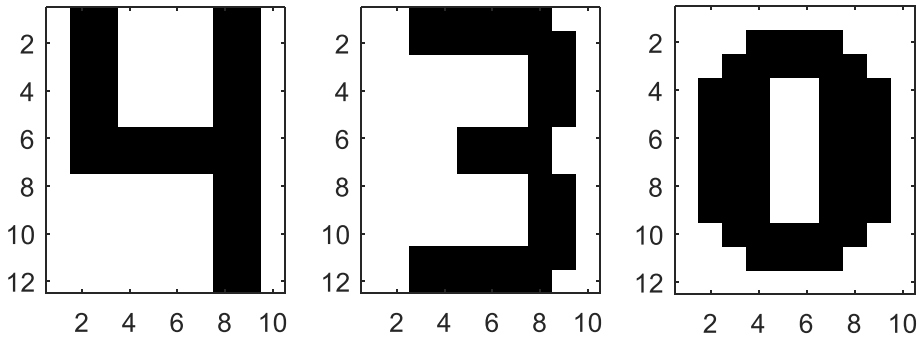
Этап 2

2. Построить сеть Хопфилда, которая будет хранить образы из заданного набора. Эталонными образами являются двоичные изображения цифр 0,1, 2, 3, 4, 6, 9 размером 12x10 на рисунке. Проверить работу сети с зашумленными образами.



2.1. Создать сеть с помощью функции *newhop*. Аттракторами построенной сети должны быть 3 образа, которые определяются вариантом задания $p = [4,3,0]$. Каждый эталонный образ задается матрицей. Цветам точек соответствуют -1 и 1. Для синтеза сети необходимо объединить эталонные образы по формуле *T* = [*p1*(:), *p2*(:), *p3*(:)].





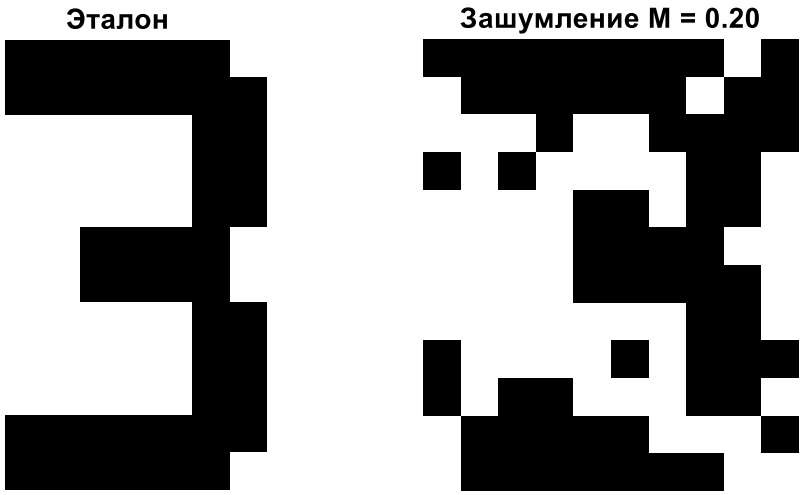
2.2. Подать в сеть первый образ, рассчитать выход сети. Число итераций задать равным 600. Результат распознавания занести в отчет.

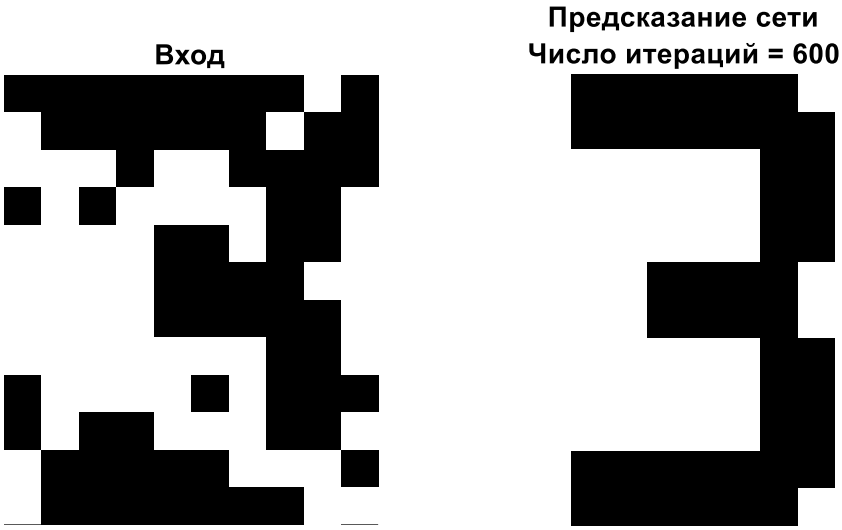
2.3. Произвести зашумление второго образа на 20%, полученный образ занести в отчет. Рассчитать выход сети. Результат распознавания занести в отчет.

Зашумление произвести следующим образом: для каждой точки изображения изменить цвет по правилу

if $r_{ij} < M$ *then инвертировать цвет точки*

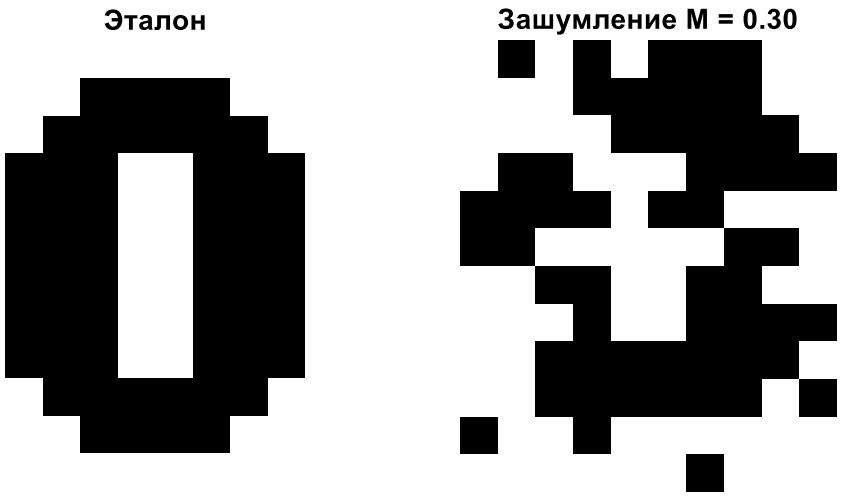
где M — степень зашумления, r —реализация случайной величины, распределенной по равномерному закону (функция *rand*).

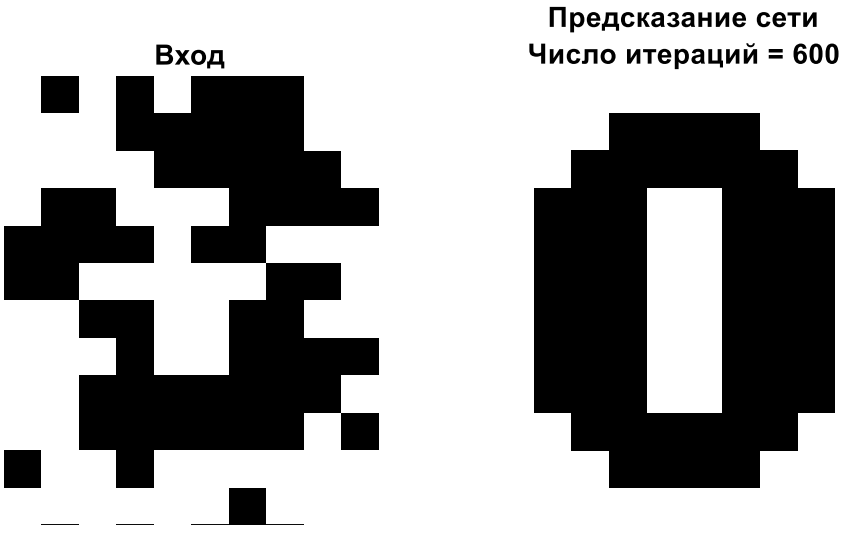




2.4. Произвести зашумление третьего образа на 30%, полученный образ занести в отчет. Рассчитать выход сети. Число итераций задать равным 600. Если необходимо, то произвести обучение несколько раз. Если

результаты распознавания неудовлетворительные, то увеличить число итераций. Результат распознавания занести в отчет.





Этап 3

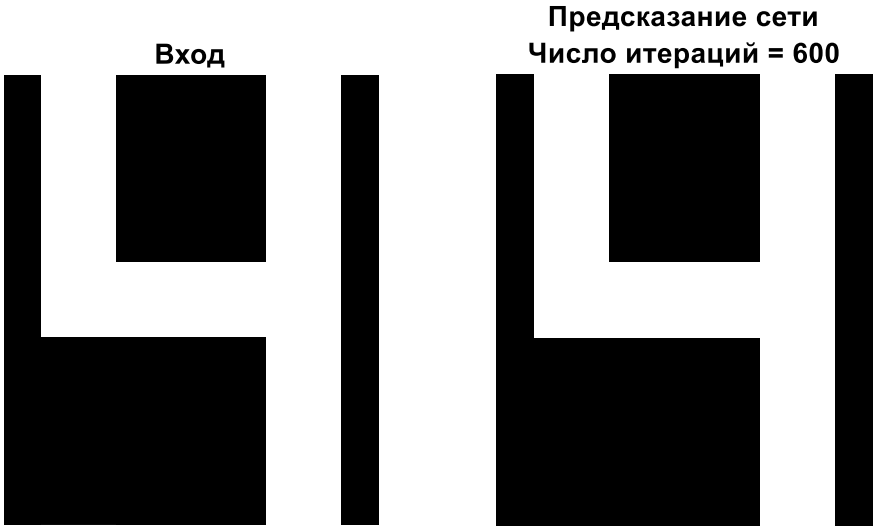
3. Построить сеть Хэмминга, которая будет хранить образы из заданного набора $p = [4,3,0]$. Эталонными образами являются двоичные изображения цифр 0,1, 2, 3, 4, 6, 9 размером 12x10. Проверить работу сети с зашумленными образами.
   3.1. Реализовать сеть Хэмминга. Сеть Хэмминга является двухслойной сеть прямого распространения. Функционирование сети производится в соответствии с правилами:

$$IW = \begin{pmatrix} p_1^T \\ \vdots \\ p_Q^T \end{pmatrix}, \qquad b^1 = \begin{pmatrix} R \\ \vdots \\ R \end{pmatrix}, \qquad a^1 = IW * p + b^1,$$

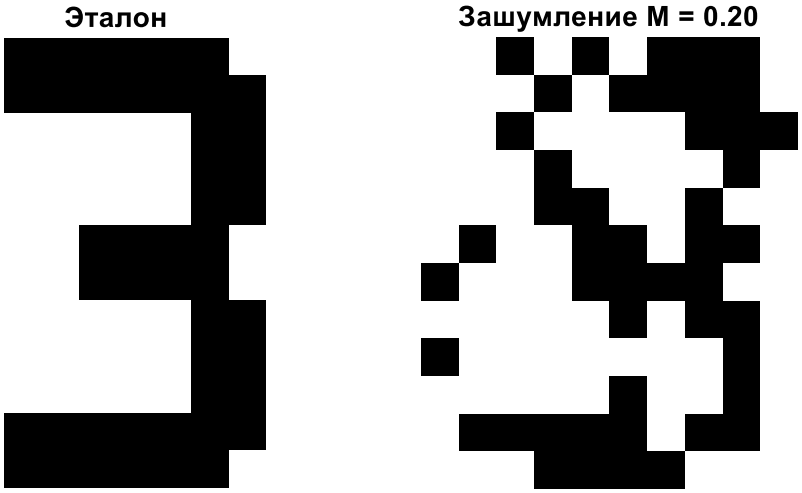
$$LW = \begin{pmatrix} 1 & -\varepsilon & \cdots & -\varepsilon \\ -\varepsilon & 1 & \cdots & -\varepsilon \\ \vdots & \vdots & \ddots & \vdots \\ -\varepsilon & -\varepsilon & \cdots & 1 \end{pmatrix}, \qquad a^2(k) = poslin\big(LW * a^2(k-1)\big),$$

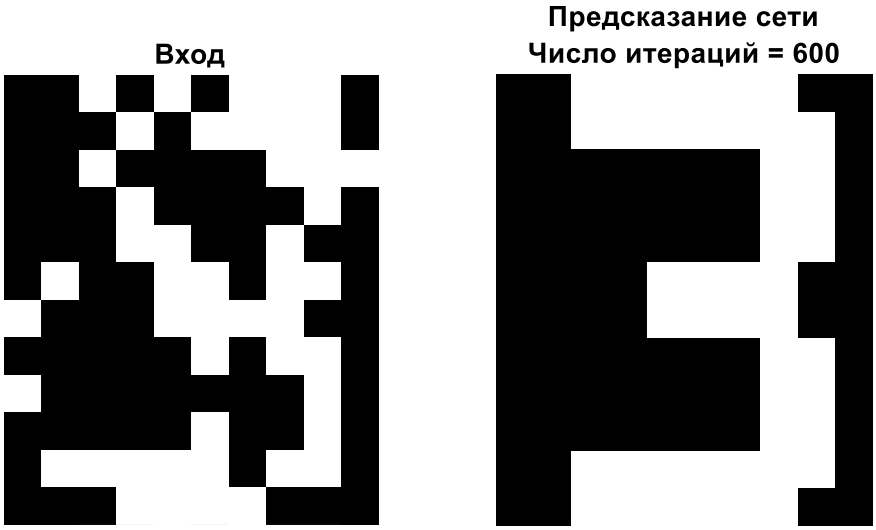
где Q – число эталонных образов, $\varepsilon = \frac{1}{Q-1}$, R – размерность входного вектора.

   3.2. Первый слой вычисляет расстояние Хэмминга между входным и эталонными векторами. Вычисления, проводимые в первом слое, реализовать по приведенному правилу.
   3.3. Для реализации работы второго слоя использовать сеть Хопфилда. Создать сеть с помощью функции *newhop*(a^1). Использовать *poslin* в качестве активационной функции (*net.layers{1}.transferFcn*). Весовые коэффициенты и смещения (LW^{11}, b^1) задать по приведенным правилам.
   3.4. Подать в сеть первый образ. Число итераций задать равным 600 и рассчитать выход сети. В результате работы сети в выходном векторе должна быть одна ненулевая компонента. Если ненулевых компонент несколько, то выбрать наибольшую компоненту. Индекс этой компоненты соответствует строке матрицы *IW*, содержащей эталонный образ. Занести выход сети и номер образа в отчет.

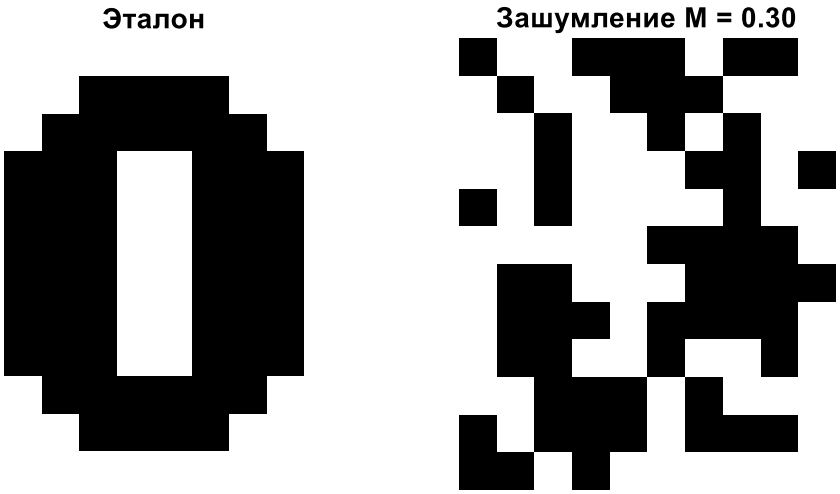


3.5. Рассчитать выход сети для зашумленного на 20% образа из Этапа 2. Занести выход сети и номер образа в отчет.

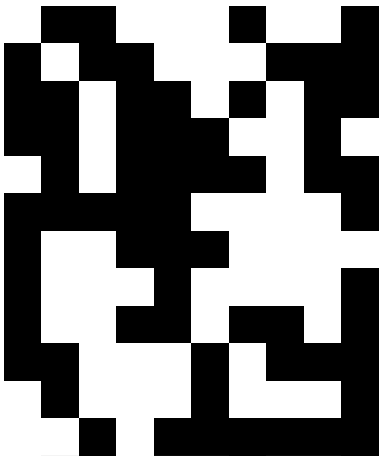




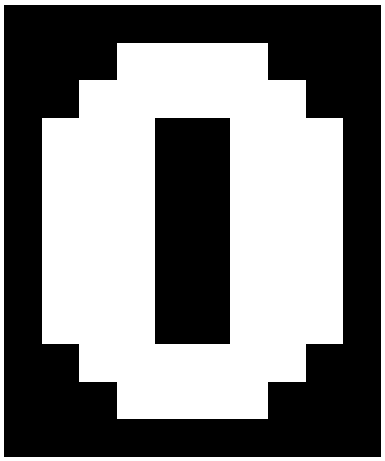
3.6. Рассчитать выход сети для зашумленного на 30% образа из Этапа 2. Занести выход сети и номер образа в отчет.



Вход



Предсказание сети
Число итераций = 600



Код программы

mat2im.m

```
function im = mat2im(mat)
    [m, n] = size(mat);

    mat_scaled = (mat + 1) / 2;

    im = zeros(m, n, 3);

    for y = 1:m
        for x = 1:n
            im(y, x, :) = [mat_scaled(y, x) mat_scaled(y, x) mat_scaled(y, x)];
            im(y, x, :) = [mat_scaled(y, x) mat_scaled(y, x) mat_scaled(y, x)];
            im(y, x, :) = [mat_scaled(y, x) mat_scaled(y, x) mat_scaled(y, x)];
        end
    end

end
```

main.m

```
% ЛР5
% Вариант 10

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

%% построение множества точек

k = 0:0.025:1;
p4 = sin(4 * pi * k);
t1 = -ones(1, length(p4));
k = 0.46:0.025:3.01;
p2 = sin(-3*k.^2+5*k+10)+0.8;
t2 = ones(1, length(p2));

r1 = 0;
r2 = 1;
r3 = 2;

X = [repmat(p4, 1, r1), p2, repmat(p4, 1, r2), p2, repmat(p4, 1, r3), p2];
y = [repmat(t1, 1, r1), t2, repmat(t1, 1, r2), t2, repmat(t1, 1, r3), t2];
yc = zeros(length(y), 3);

for i = 1:length(y)
    if y(i) == -1
        yc(i, :) = [1 0 0];
    else
        yc(i, :) = [0 0 1];
    end
end
x = 0:0.025:0.025 * (length(X)-1);

plot(x, X, 'green');
hold on
scatter(x, X, 10, yc, 'filled');
grid on;
```

```
axis([x(1) x(end) -2 2]);

X = con2seq(X);
y = con2seq(y);

%% создание сети

net = layrecnet(1:2, 8, 'trainoss');
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'tansig';
net = configure(net, X, y);

[Xs, Xi, Ai, ys] = preparets(net, X, y);

%% Обучение

net.trainParam.epochs = 200;
net.trainParam.goal = 1e-5;

net = train(net, Xs, ys, Xi, Ai);

%% Выход сети

plot(x, cell2mat(X), 'green');
hold on

scatter(x, cell2mat(X), 20, cell2mat(sim(net, X)), 'filled');
colorbar

axis([x(1) x(end) -2 2]);
grid on

%% Результаты в сравнении с эталонами

plot(x, cell2mat(X), 'green');
hold on
scatter(x, cell2mat(X), 20, yc);

y_pred = (cell2mat(sim(net, X)) >= 0) * 1 + (cell2mat(sim(net, X)) <= 0) * (-
1);

fprintf('Правильно классифицировано точек %d/%d\n', ...
        sum(y_pred == cell2mat(y)), length(y));

yc_pred = zeros(length(y), 3);

for i = 1:length(y)
    if y_pred(i) == -1
        yc_pred(i, :) = [1 0 0];
    else
        yc_pred(i, :) = [0 0 1];
    end
end

scatter(x, cell2mat(X), 9, yc_pred, 'filled');
```

```
legend('Сигнал', 'Эталон', 'Предсказание сети');

grid on;
axis([x(1) x(end) -2 2]);

%% Цифры

% 4
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];

% 3
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];

% 0
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];

[m, n] = size(p1); % 12 x 10

im1 = mat2im(p1);
im2 = mat2im(p2);
im3 = mat2im(p3);
```

```
subplot(1, 3, 1)
image(~im1);
%axis off
axis image

subplot(1, 3, 2)
image(~im2);
%axis off
axis image

subplot(1, 3, 3)
image(~im3);
%axis off
axis image

%% Сеть Хопфилда

p0 = [p1(:) p2(:) p3(:)];

net = newhop(p0);

%% Зашумление

M = 0.3;

p_et = p3;
p_noise = p_et;

for y = 1:m
    for x = 1:n
        if rand() < M
            if p_noise(y, x) == 1
                p_noise(y, x) = -1;
            else
                p_noise(y, x) = 1;
            end
        end
    end
end

im_et = mat2im(p_et);
im_noise = mat2im(p_noise);

subplot(1, 2, 1)
image(~im_et);
title('Эталон');
axis off
axis image

subplot(1, 2, 2)
image(~im_noise);
title(sprintf('Зашумление M = %.2f', M));
axis off
axis image
```

```
%% Предсказание

n_iter = 600;
X = {p_noise(:)};

[Y, ~, ~] = sim(net, {1 n_iter}, {}, X);

%% Вывод результата

Y_res = reshape(Y{end}, [m n]);
X_mat = reshape(cell2mat(X), [m n]);

im_X = mat2im(X_mat);
im_pred = mat2im(Y_res);

subplot(1, 2, 1)
image(~im_X);
title('Вход');
axis off
axis image
box on

subplot(1, 2, 2)
image(~im_pred);
title(sprintf('Предсказание сети\nЧисло итераций = %d', n_iter));
axis off
axis image

%% Сеть Хэмминга

p0 = p_noise(:);

IW = [p1(:)'; p2(:)'; p3(:)'];
eps = 1 / (3 - 1);
LW = [   1 -eps -eps;
      -eps    1 -eps;
      -eps -eps    1];
b = [3; 3; 3];

a1 = IW * p0 + b;

net = newhop(a1);
net.layers{1}.transferFcn = 'poslin';
net.b{1} = b;
net.LW{1, 1} = LW;

n_iter = 600;
[Y, ~, ~] = sim(net,{1 n_iter}, {}, {a1});

%% Вывод результата

[~, ind] = max(Y{end});

Y_res = IW(ind, :)';
```

```
Y_res = reshape(Y_res, [m n]);
X_mat = reshape(p0, [m n]);

im_X = mat2im(X_mat);
im_pred = mat2im(Y_res);

subplot(1, 2, 1)
image(im_X);
title('Вход');
axis off
axis image

subplot(1, 2, 2)
image(im_pred);
title(sprintf('Предсказание сети\nЧисло итераций = %d', n_iter));
axis off
axis image
```

Выводы

В лабораторной работе было проведено исследование свойств сетей Хопфилда, Хэмминга и Элмана, алгоритмов обучения, а также применения сетей в задачах распознавания статических и динамических образов.