**Лабораторная работа № 6**

«Сети Кохонена».

Постановка задачи.

Целью работы является исследование свойств слоя Кохонена, карты Кохонена, а также сетей векторного квантования, обучаемых с учителем, алгоритмов обучения, а также применение сетей в задачах кластеризации и классификации.

Ход работы.

1. Использовать слой Кохонена для выполнения кластеризации множества точек. Проверить качество разбиения.

2. Использовать карту Кохонена для выполнения кластеризации множества точек.

3. Использовать карту Кохонена для нахождения одного из решений задачи коммивояжера.

4. Использовать сеть векторного квантования, обучаемую с учителем, (LVQ-сеть) для классификации точек в случае, когда классы не являются линейно разделимыми.

Теория.

Нейронные сети Кохонена — класс нейронных сетей, основным элементом которых является слой Кохонена. Слой Кохонена состоит из адаптивных линейных сумматоров. Как правило, выходные сигналы слоя Кохонена обрабатываются по правилу «Всё или ничего»: наибольший сигнал превращается в единичный, остальные обращаются в ноль.

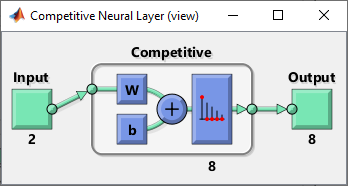
Сети Кохонена различают по способам настройки входных весов сумматоров и по решаемым ими задачам. Наиболее известные из них:

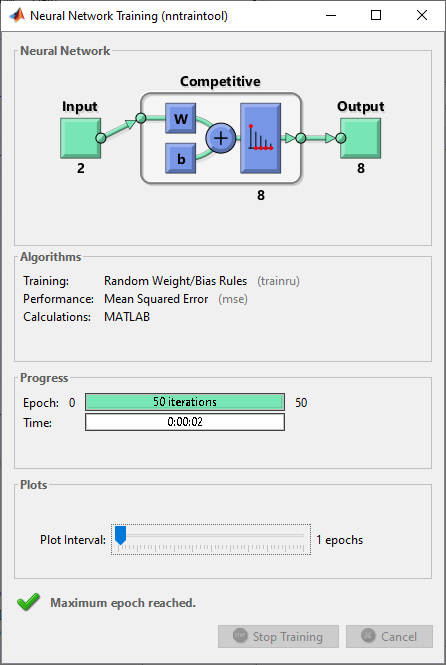
* Сети векторного квантования сигналов, тесно связанные с простейшим базовым алгоритмом кластерного анализа (метод динамических ядер или K-средних).
* Самоорганизующиеся карты Кохонена.
* Сети векторного квантования, обучаемые с учителем.

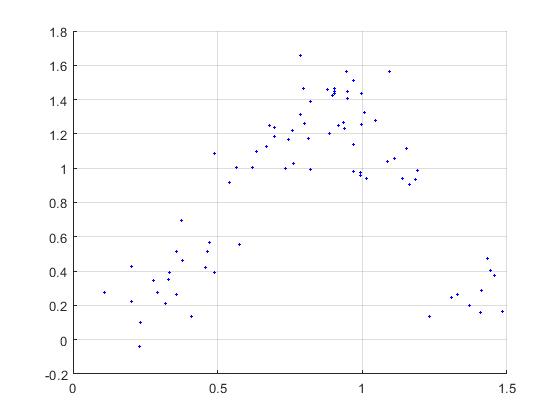
Входные данные и результаты.

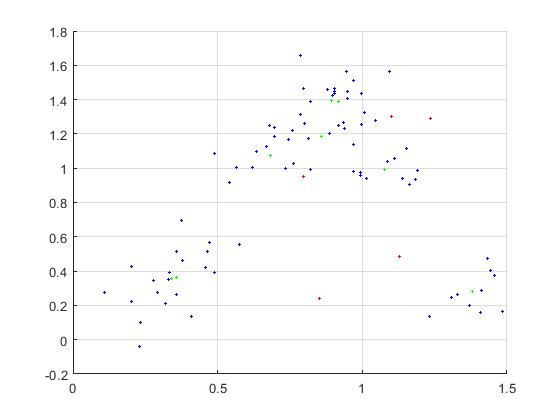
Вариант № 15.

Первое задание

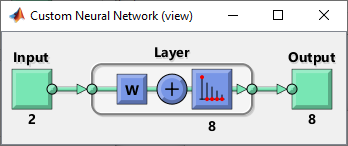


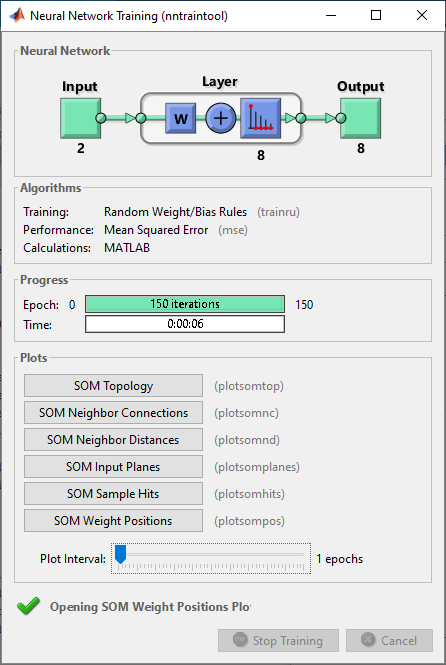


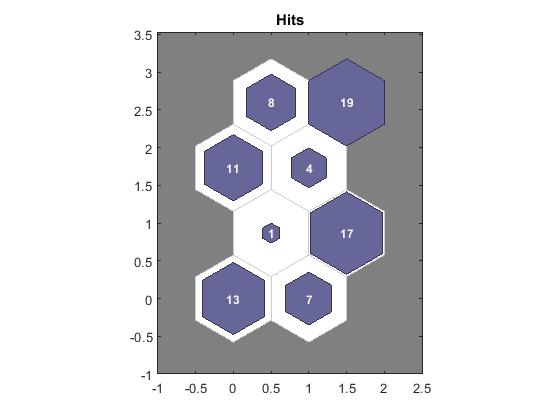


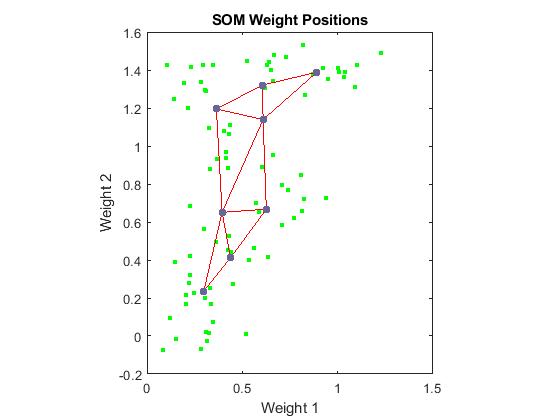


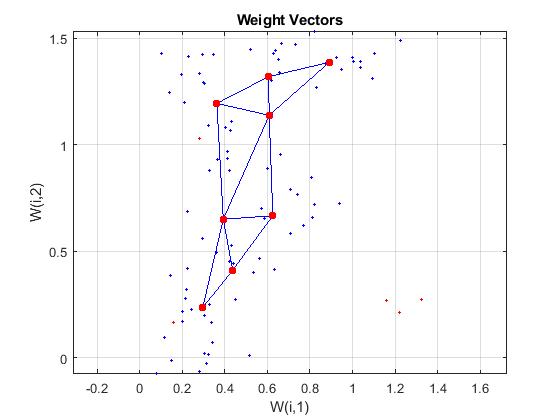
Второе задание



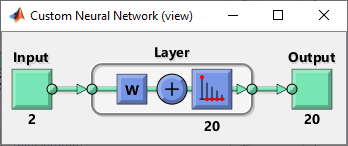


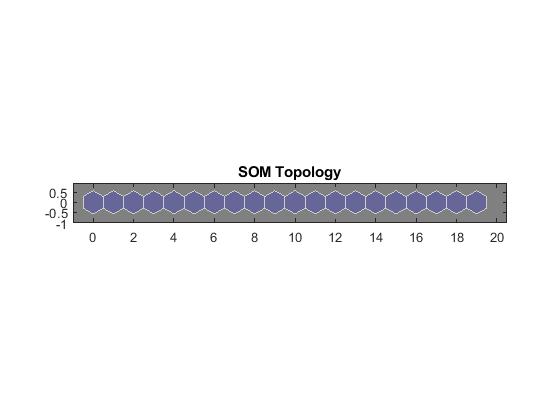


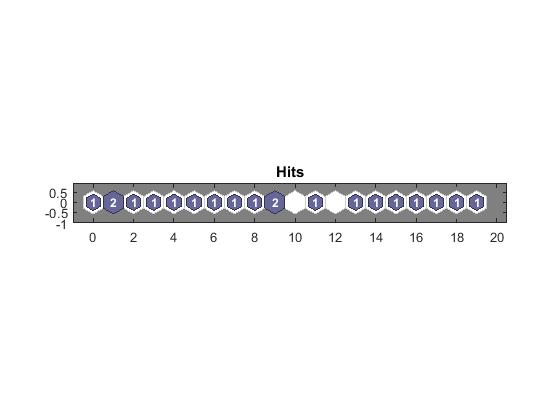


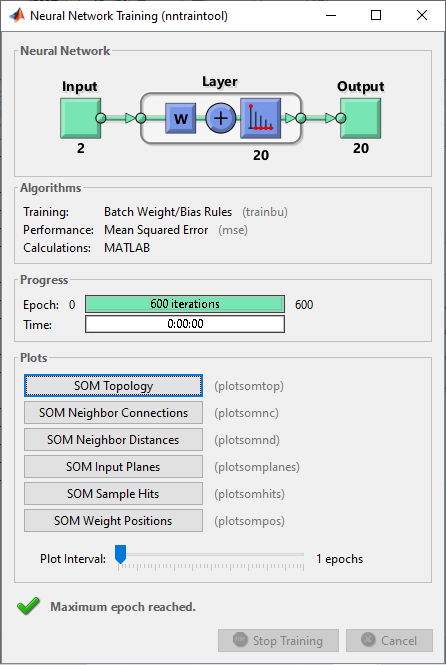


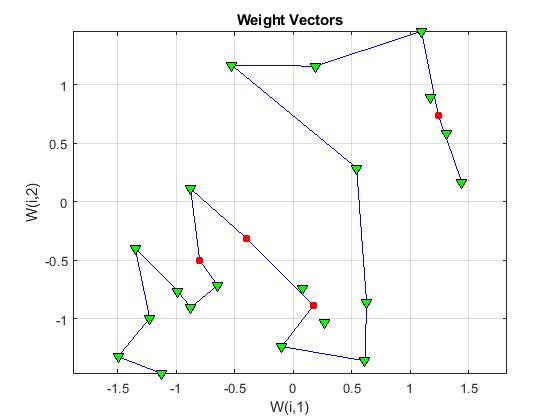
Третье задание

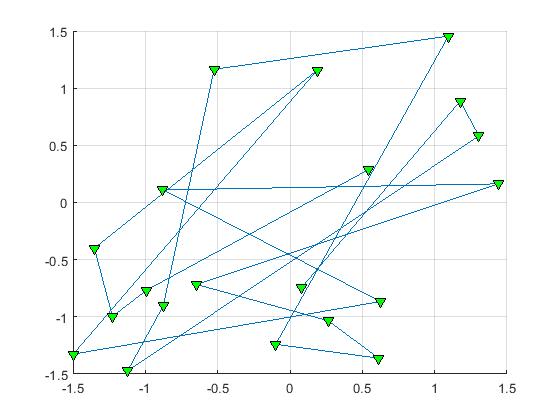




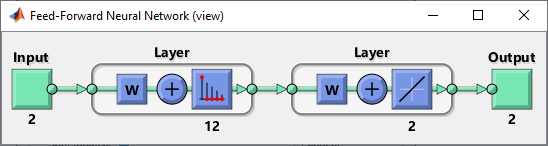


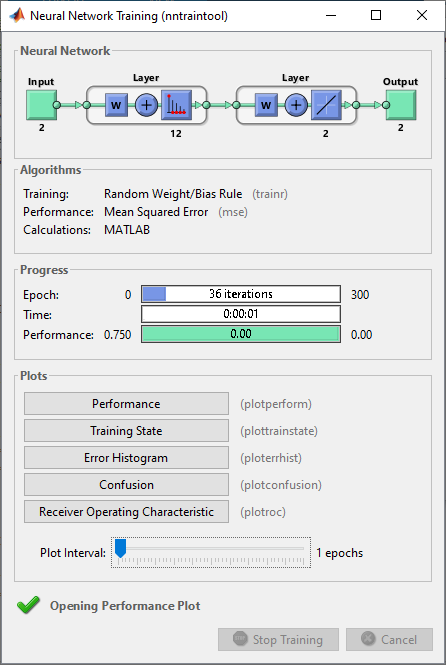


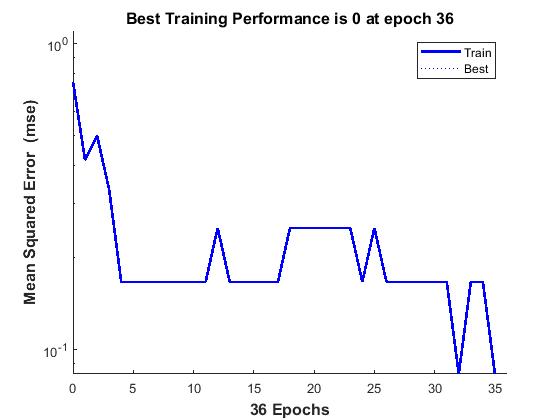


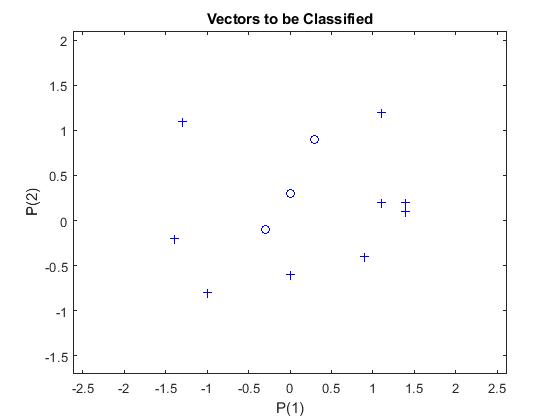


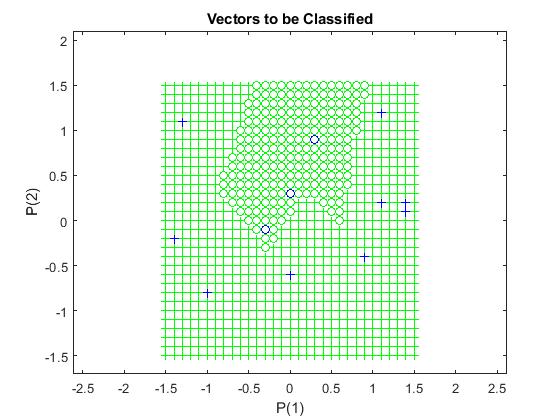
Четвертое задание











Исходный код.

clear;

clc;

X = [0, 1.5; 0, 1.5];

clusters = 8;

points = 10;

deviation = 0.1;

P = nngenc(X, clusters, points, deviation);

figure;

hold on;

grid on;

scatter(P(1, :), P(2, :), 5, [0 0 1], 'filled');

net = competlayer(8);

net = configure(net, P);

view(net);

net.trainParam.epochs = 50;

net = train(net, P);

RandomPoints = 1.5 \* rand(2, 5);

res = vec2ind(sim(net, RandomPoints));

figure;

hold on;

grid on;

scatter(P(1, :), P(2, :), 5, [0 0 1], 'filled');

scatter(net.IW{1}(:, 1), net.IW{1}(:, 2), 5, [0 1 0], 'filled');

scatter(RandomPoints(1, :), RandomPoints(2, :), 5, [1 0 0], 'filled');

clear;

clc;

X = [0, 1.5; 0, 1.5];

clusters = 8;

points = 10;

deviation = 0.1;

P = nngenc(X, clusters, points, deviation);

net = newsom(X, [2 4]);

net = configure(net, X);

view(net);

net.inputWeights{1,1}.learnParam.init\_neighborhood = 3;

net.inputWeights{1,1}.learnParam.steps = 100;

net.trainParam.epochs = 150;

net = train(net, P);

RandomPoints = 1.5 \* rand(2, 5);

res = vec2ind(sim(net, RandomPoints));

figure;

hold on;

grid on;

scatter(P(1, :), P(2, :), 5, [0 0 1], 'filled');

scatter(net.IW{1}(:, 1), net.IW{1}(:, 2), 5, [0 1 0], 'filled');

scatter(RandomPoints(1, :), RandomPoints(2, :), 5, [1 0 0], 'filled');

plotsom(net.IW{1, 1}, net.layers{1}.distances);

clear;

clc;

T = -1.5 \* ones(2, 20) + 3 \* rand(2, 20);

figure;

hold on;

grid on;

plot(T(1,:), T(2,:), '-V', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g', 'MarkerSize', 7);

net = newsom(T, 20);

net = configure(net, T);

view(net);

net.trainParam.epochs = 600;

net = train(net, T);

figure;

hold on;

grid on;

plotsom(net.IW{1,1}, net.layers{1}.distances);

plot(T(1,:), T(2,:), 'V', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g', 'MarkerSize', 7);

clear;

clc;

P = [ 1.4  1.4    0 -0.3 -1.3 -1.4  0.9 0.3  1.1   -1   1.1    0;

      0.2  0.1 -0.6 -0.1  1.1 -0.2 -0.4 0.9  1.2 -0.8   0.2  0.3];

T = [1 1 1 -1 1 1 1 -1 1 1 1 -1];

plotpv(P, max(0, T));

Ti = T;

Ti(Ti == 1) = 2;

Ti(Ti == -1) = 1;

Ti = ind2vec(Ti);

portion = [nnz(T(T == -1)) nnz(T(T == 1))] / numel(T);

net = lvqnet(12, 0.1);

net = configure(net, P, Ti);

view(net)

net.IW{1,1}

net.LW{2,1}

net.trainParam.epochs = 300;

net = train(net, P, Ti);

[X,Y] = meshgrid([-1.5 : 0.1 : 1.5], [-1.5 : 0.1 : 1.5]);

res = sim(net, [X(:)'; Y(:)']);

res = vec2ind(res) - 1;

figure;

plotpv([X(:)'; Y(:)'], res);

point = findobj(gca,'type','line');

set(point,'Color','g');

hold on;

plotpv(P, max(0, T));

Выводы:

Самоорганизующиеся карты Кохонена служат, в первую очередь, для визуализации и первоначального анализа данных. Каждая точка данных отображается соответствующим кодовым вектором из многомерной сетки. Так получают представление данных на плоскости - «Карту данных».

Из минусов слоев Кохонена стоит отметить то, что нам нужно заранее знать количество кластеров. Получается, мы сильно зависим от этой величины. Также не совсем очевидно, что делать в ситуации, когда точка одинаково удалена до нескольких центров классов? Также интересен вопрос, а как проверить качество обучения?

Несмотря на некоторые недостатки сетей Кохонена, люди до сих пор применяют их для решения узкоспециализированных задач «Нейроинформатики».