

Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 6
по курсу «Нейроинформатика»
Тема: Сети Кохонена.

Студент: Куликов А.В.
Группа: М80-408Б-17
Преподаватель: Аносова Н.П.
Дата: 18 декабря 2020
Оценка:

Цель работы: 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.

Основные этапы 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.

Оборудование:

Процессор: AMD Ryzen 5 Mobile 3550H

Объем оперативной памяти: 8 Гб

Программное обеспечение:

Python 3.8.5, MATLAB r2020

Сценарий выполнения работы:

Задание №1

```
clear;
clc;

% Формирование множества точек
X = [0 1.5;
     0 1.5];
clusters = 8;
points = 10;
deviation = 0.1;
P = nngenc(X, clusters, points, deviation);

% Создание и конфигурация сети
net = competlayer(8);
net = configure(net, P);
view(net);
net.divideFcn = '';

% Обучение сети
net.trainParam.epochs = 50;
net = train(net, P);

disp("First layer weights:");
```

```

disp(net.IW{1});

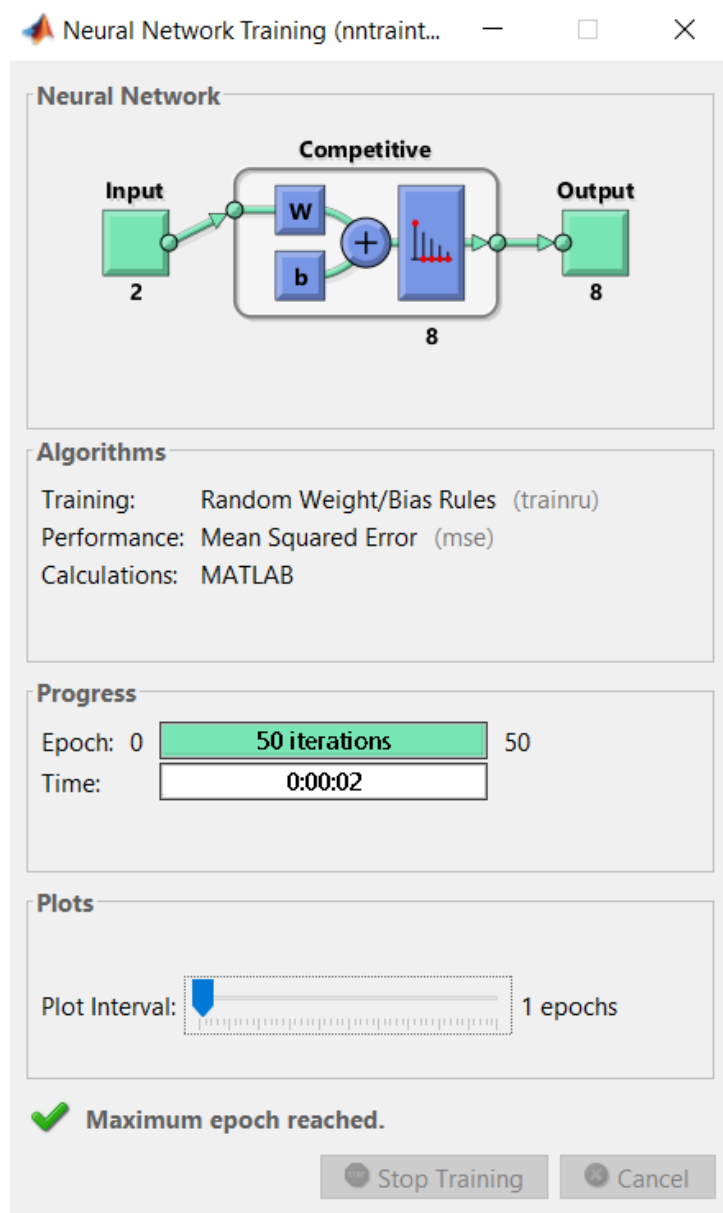
% Проверка качества
sample = 1.5*rand(2, 5);
result = vec2ind(sim(net, sample));

% Результат кластеризации тестового множества
disp("Result:");
disp(result);

figure;
hold on;
grid on;
scatter(P(1, :), P(2, :), 5, [0 1 0], 'filled');
scatter(net.IW{1}(:, 1), net.IW{1}(:, 2), 5, [0 0 1], 'filled');
scatter(sample(1, :), sample(2, :), 5, [1 0 0], 'filled');

```

Структура сети

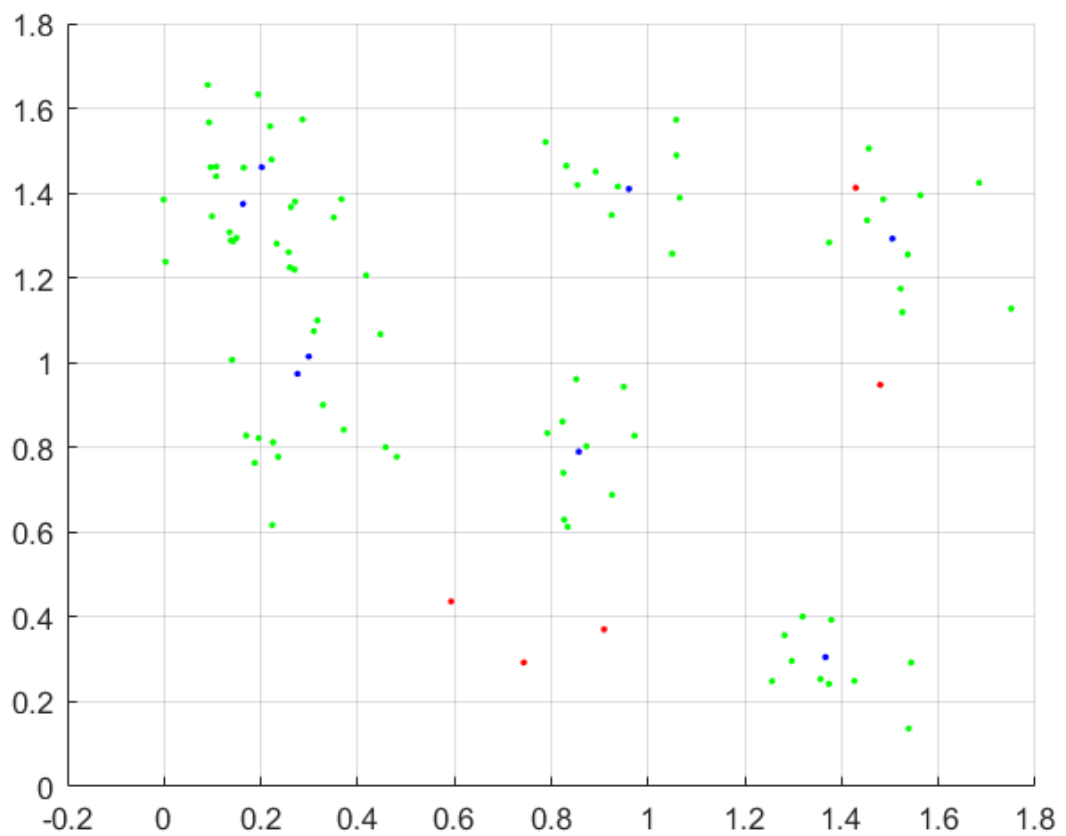


Веса сети:

First layer weights:

0.2757	0.9738
0.9604	1.4096
1.3666	0.3053
0.1629	1.3746
1.5050	1.2927
0.2017	1.4611
0.8569	0.7899
0.2990	1.0144

Результат обучения сети и выход сети для тестового множества.



Красные точки – тестовые. Зеленые – исходное множество для кластеризации. Синие – центры кластеров.

Выход сети для тестового множества:

Result:

5	7	7	7	7
---	---	---	---	---

Задание №2

```
clear;
clc;

% Формирование множества точек
X = [0 1.5;
     0 1.5];
clusters = 8;
points = 10;
deviation = 0.1;
P = nngenc(X, clusters, points, deviation);

% Создание и конфигурация сети
net = newsom(X, [2 4]);
net = configure(net, X);
net.trainParam.epochs = 150;

% Обучение сети
net.divideFcn = '';
net = train(net, P);

figure;
plotsomhits(net,P)
figure;
plotsompos(net,P)

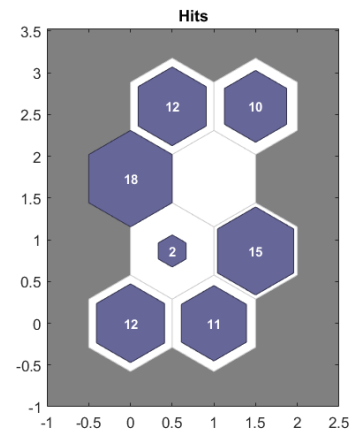
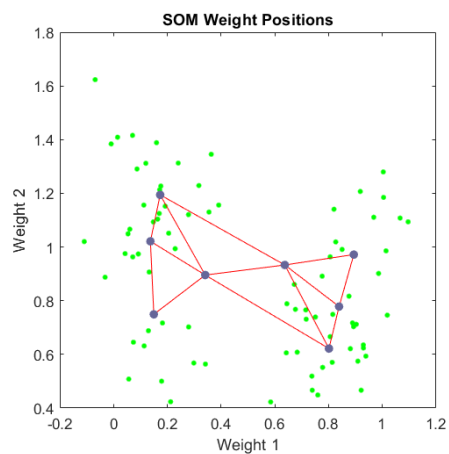
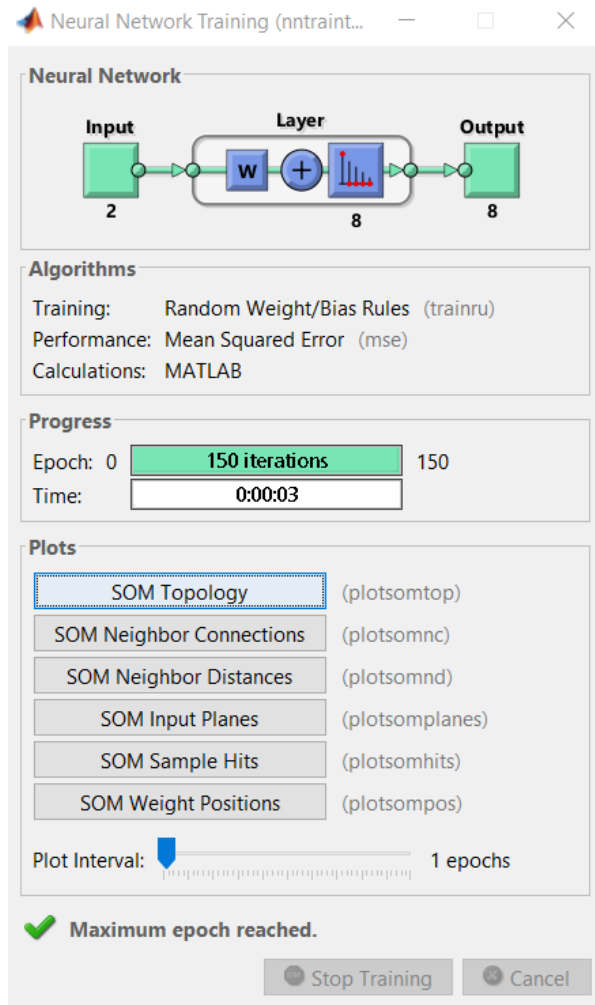
disp("First layer weights:");
disp(net.IW{1});

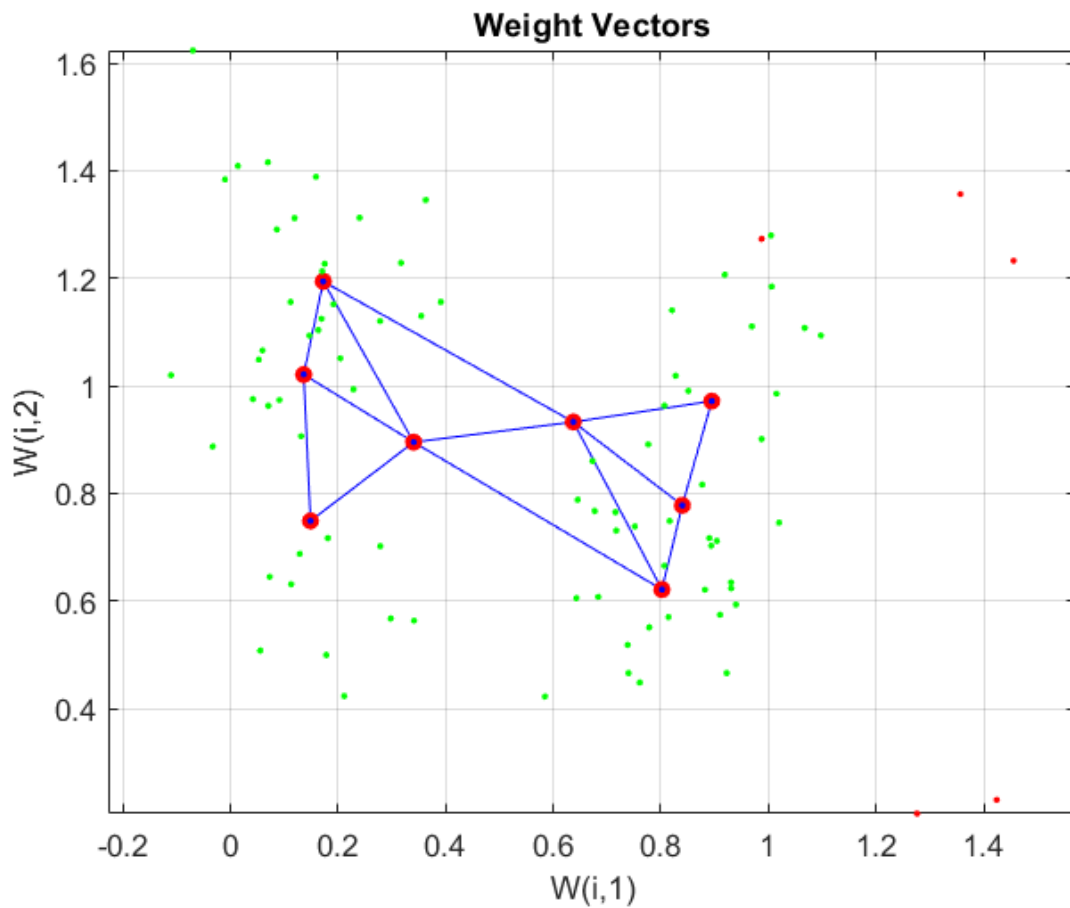
% Проверка качества разбиения
sample = 1.5 * rand(2, 5);
result = vec2ind(sim(net, sample));

disp("Result:");
disp(result);

figure;
hold on;
grid on;
plotsom(net.IW{1,1},net.layers{1}.distances);
scatter(P(1, :), P(2, :), 5, [0 1 0], 'filled');
scatter(net.IW{1}(:, 1), net.IW{1}(:, 2), 5, [0 0 1], 'filled');
scatter(sample(1, :), sample(2, :), 5, [1 0 0], 'filled');
```

Структура сети





Зеленые – исходное множество для кластеризации. Красно-синие – центры кластеров.

Веса сети:

First layer weights:

0.8947 0.9718

0.8399 0.7781

0.6374 0.9330

0.8020 0.6217

0.1726 1.1943

0.3407 0.8955

0.1362 1.0212

0.1489 0.7492

Выход сети для тестового множества:

Result:

1 4 1 1 4

Задание №3

```
clear;
clc;

% Формирование множества точек
N = 20;
T = -1.5 + 3*rand(2, N);

figure;
hold on;
grid on;
plot(T(1,:), T(2,:), '-v', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'g',
'MarkerSize', 7);

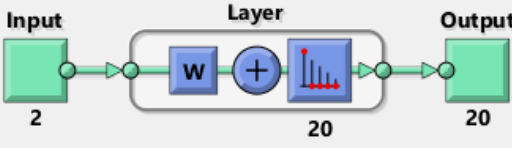
% Создание и конфигурация сети
net = newsom(T, N);
net = configure(net, T);
net.divideFcn = '';
net.trainParam.epochs = 15000;
net = train(net, T);

% Отображение координат городов и центров кластеров
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);
hold off;
```


Структура сети

Neural Network Training (nntraint...

Neural Network



Algorithms

Training: Batch Weight/Bias Rules (trainbu)
Performance: Mean Squared Error (mse)
Calculations: MATLAB

Progress

Epoch: 0 **20000 iterations** 20000
Time: 0:00:24

Plots

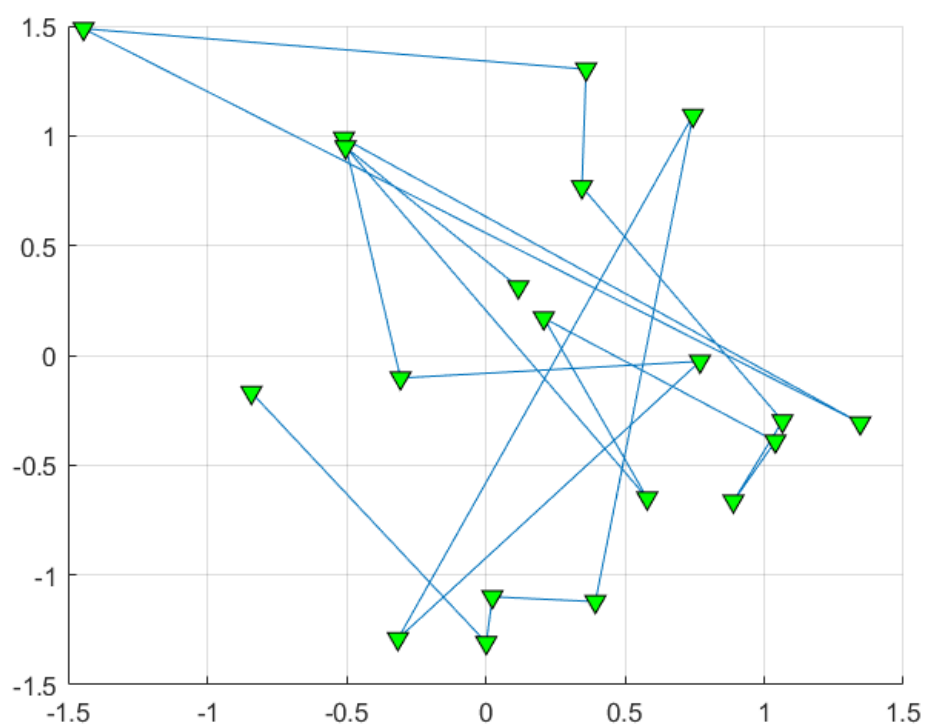
SOM Topology	(plotsomtop)
SOM Neighbor Connections	(plotsomnc)
SOM Neighbor Distances	(plotsomnd)
SOM Input Planes	(plotsomplanes)
SOM Sample Hits	(plotsomhits)
SOM Weight Positions	(plotsompos)

Plot Interval: 1 epochs

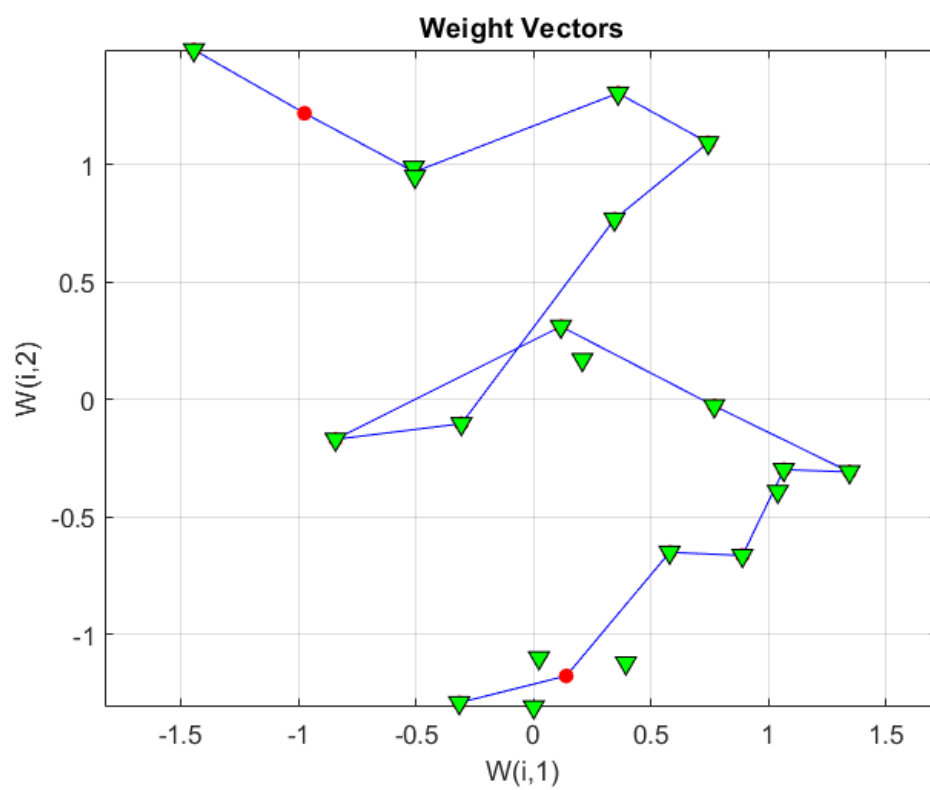
✓ **Maximum epoch reached.**

Stop Training Cancel

Исходное множество городов



Решение задачи коммивояжера (неполное)



Задание №4

```
clear;
clc;

% Задание обучающего множества
P = [0.6 0.2 1.2 0.9 0.2 -0.3 -1.1 -0.3 0.2 0.5 0.4 -1.3;
     -0.5 -1.2 1.1 -0.8 -1.5 -0.6 -1 -1.3 -0.1 0.5 -1.4 -0.6];
T = [-1 1 -1 -1 1 -1 -1 -1 1 -1 1 -1];

figure;
hold on;
grid on;
plotpv(P, max(T, 0));

Ti = T;
Ti(Ti == 1) = 2;
Ti(Ti == -1) = 1;

p_class1 = sum(Ti(:) == 1);
n_samples = size(Ti, 2);

Ti = ind2vec(Ti);

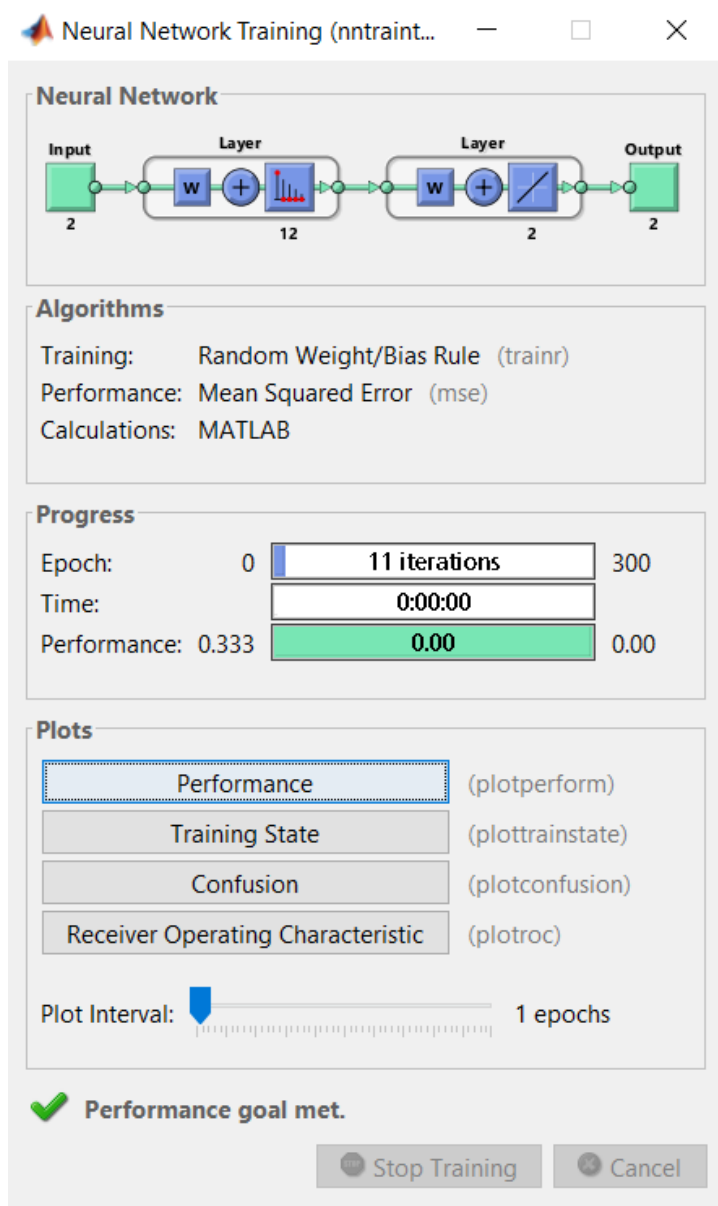
% Создание и конфигурация сети
net = newlvq(minmax(P), 12, [p_class1 / n_samples, 1 - p_class1 / n_samples], 0.1);
net = configure(net, P, Ti);
net.trainParam.epochs = 300;

% Обучение сети
net = train(net, P, Ti);
disp("Weights:");
disp("IW_1:");
disp(net.IW{1,1});
disp("IW_2:");
disp(net.LW{2,1});

% Проверка качества обучения
[X,Y] = meshgrid(-1.5 : 0.1 : 1.5, -1.5 : 0.1 : 1.5);
result = sim(net, [X(:)'; Y(:)']);
result = vec2ind(result) - 1;

figure;
plotpv([X(:)'; Y(:)'], result);
point = findobj(gca, 'type', 'line');
set(point, 'Color', 'g');
hold on;
plotpv(P, max(0, T));
```

Структура сети



Веса сети:

Weights:

IW_1:

-0.9004 -0.8623

0.6694 0.6183

-0.0950 -0.0800

0.6496 -0.4820

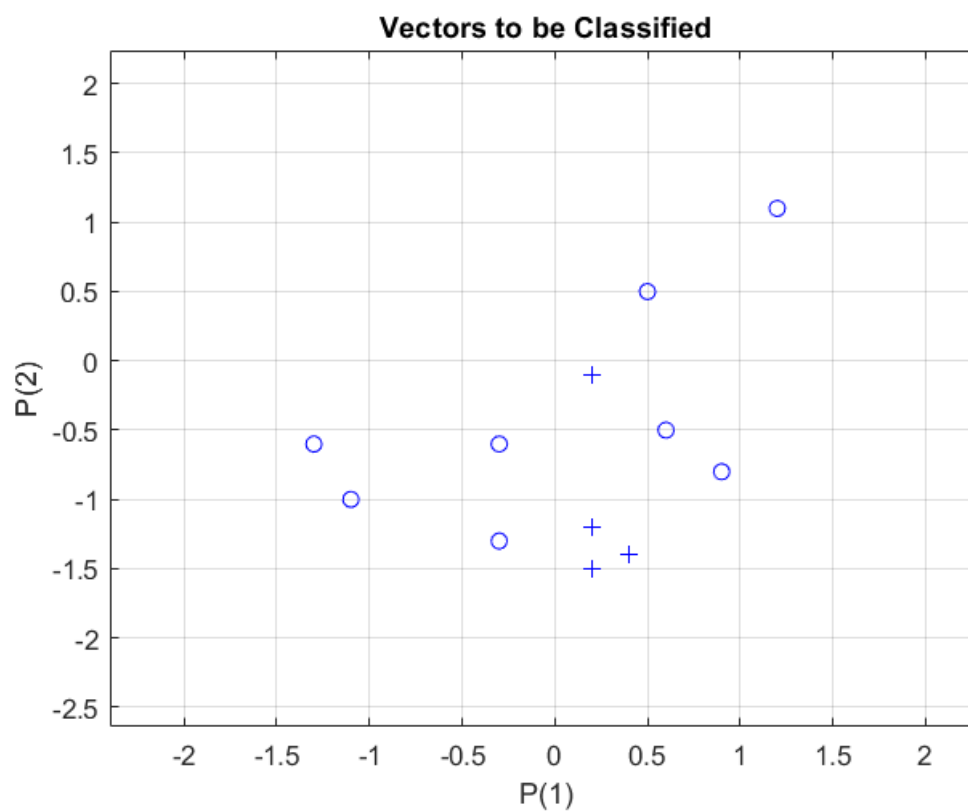
-0.0950 -0.0800

-0.0750	-0.1000
-0.1562	-0.2144
-0.0750	-0.1000
0.4002	-1.0090
0.0924	-0.1430
-0.0250	-0.1600
-0.0250	-0.1600

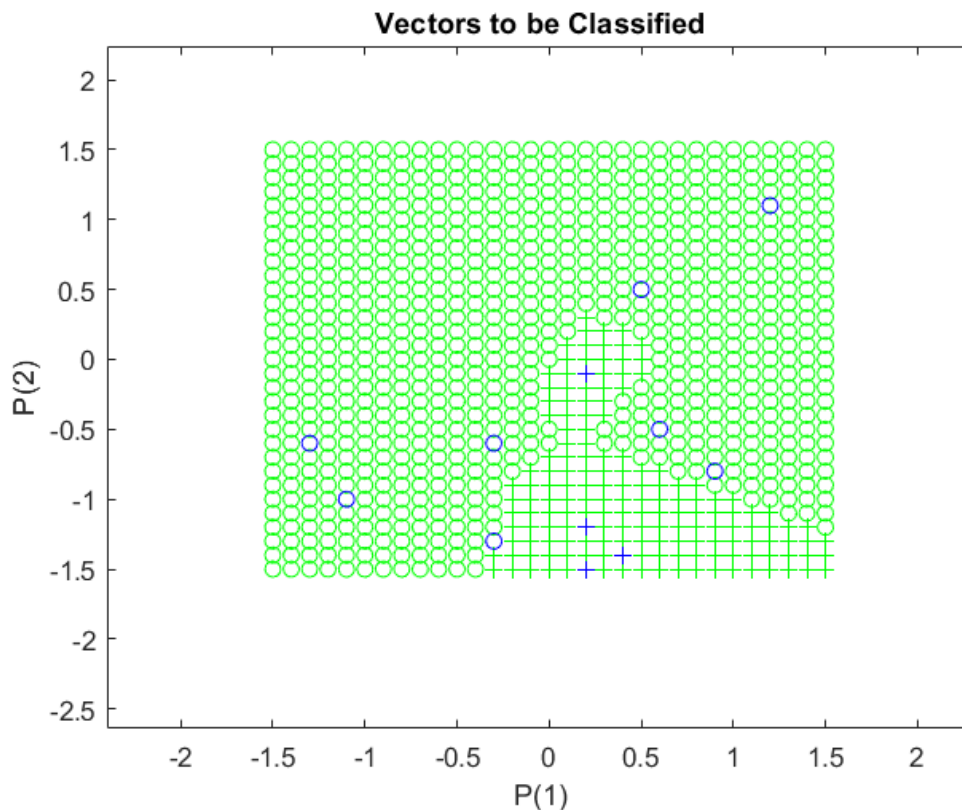
IW_2:

1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1

Обучающее множество



Результат классификации точек сетки



Выводы:

Нейронные сети Кохонена — класс нейронных сетей, основным элементом которых является слой Кохонена.

Слой Кохонена состоит из адаптивных линейных сумматоров. Количество нейронов равно количеству кластеров, количество входных переменных нейронной сети равно числу признаков.

При обучении слоя Кохонена на вход подается входной вектор и вычисляются его скалярные произведения с векторами весов, связанными со всеми нейронами Кохонена. Нейрон с максимальным значением скалярного произведения объявляется победителем, и его веса подстраиваются таким образом, что вектор весов нейрона-победителя сдвигается в направлении победившего образца.

Таким образом после обучения нейронной сети Кохонена близкие входные векторы будут активировать один и тот же нейрон Кохонена. Обучение слоя протекает без учителя. От процесса обучения требуется лишь, чтобы в результате обучения отделялись друг от друга сильно отличающиеся входные вектора.

Самоорганизующаяся карта Кохонена, в отличие от слоя Кохонена, позволяет не только кластеризовать данные, но и визуализировать результат кластеризации.

Карта Кохонена состоит из ячеек. Каждая ячейка связана с определенным выходным нейроном и представляет собой «сферу влияния» данного нейрона. Объекты, векторы признаков которых оказываются ближе к вектору весов данного нейрона, попадают в ячейку, связанную с ним.

В процессе обучения на каждой итерации выбирается один из входных векторов. Для входного вектора находится ближайший нейрон и объявляется победителем. После чего, нейрон-победитель и его соседи (т.е. те нейроны, которые находятся в его сфере влияния, те, для которых функция близости больше какого-то порогового значения) стягиваются в направлении обрабатываемого входного вектора.

Сеть векторного квантования состоит из двух слоев: соревновательный слой и линейный слой. Соревновательный слой выполняет кластеризацию входных векторов на некоторое число подклассов, а линейный слой объединяет эти подклассы в желаемое число классов.

Слой Кохонена и самоорганизующаяся карта Кохонена решают задачу кластеризации. Сети векторного квантования позволяют решать задачу дискретной аппроксимации.