МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра 806: «Вычислительная математика и программирование»

Лабораторная работа № 6
по курсу «Нейроинформатика»

Студент: А. О. Тояков
Преподаватель: Н. П. Аносова
Группа: М8о-407б-18
Дата:
Оценка:
Подпись:

Москва, 2021

СЕТИ С ОБРАТНЫМИ СВЯЗЯМИ

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.

Вариант № 26:

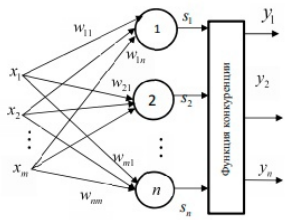
$$\begin{bmatrix} 0.9 & 0.6 & 0.6 & 0 & 0.8 & -0.8 & 0.3 & -1.2 & -0.7 & -0.3 & 1.3 & -1.4 \\ 0.8 & -0.1 & 1.3 & -0.6 & -0.8 & -0.2 & 0.3 & -1.2 & -0.7 & 0.9 & -0.4 & 0.2 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

СТРУКТУРА МОДЕЛЕЙ

Нейронная сеть из слоя Кохонена

Слой Кохонена - стандартный линейный слой, состоящий из сумматоров с линейной функции активации. Однако выходом этого слоя является 1 в случае если нейрон обладает максимальным значением сигнала среди всех остальных сигналов, и 0 для всех остальных нейронов.

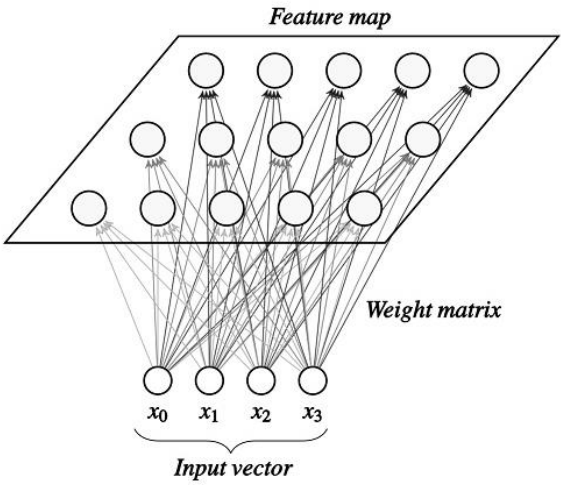


Обучить такой слой можно итеративно или с помощью алгоритма кластеризации k-means. То есть необходимо подбирать веса нейронов w так, чтобы они представляли собой центры тяжести c_j для каждого из кластеров K_j, на которые слой Кохонена делит входное множество X.

Такое обучение корректно, поскольку скалярное произведение <x, w> пропорционально косинусу угла между векторами, а значит максимальный сигнал будет у нейрона, чьи веса w наиболее похожи на входные данные x, которые активируют этот нейрон.

Самоорганизующиеся карты Кохонена

Этот вид нейронных сетей принципиально похож на слой Кохонена, однако при этом сохраняет топологическую информацию о кластерах и при обучении корректируется не только активированый нейрон, но и соседние.



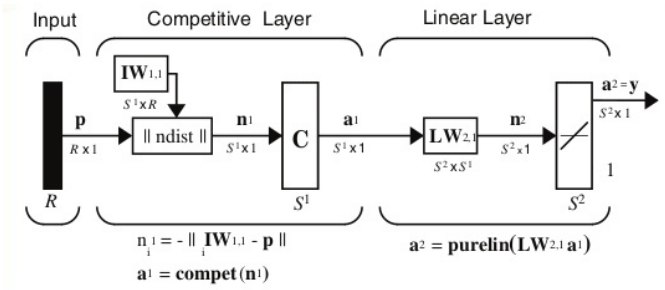
Этот алгоритм позволяет понижать размерность данных, а также служит для визуализации.

Сеть векторного квантования с учителем (LVQ Network)

Эта сеть решает задачу классификации объектов, но при этом выделяя подклассы заданных классов.

Сеть состоит из 2 слоев:

1) Слой Кохонена, определяющая кластеры из входных данных. Результаты работы этой сети будут интерпретироваться как подклассы.
2) Линейный слой, который определяет принадлежность каждого из кластеров заданному классу.



ХОД РАБОТЫ

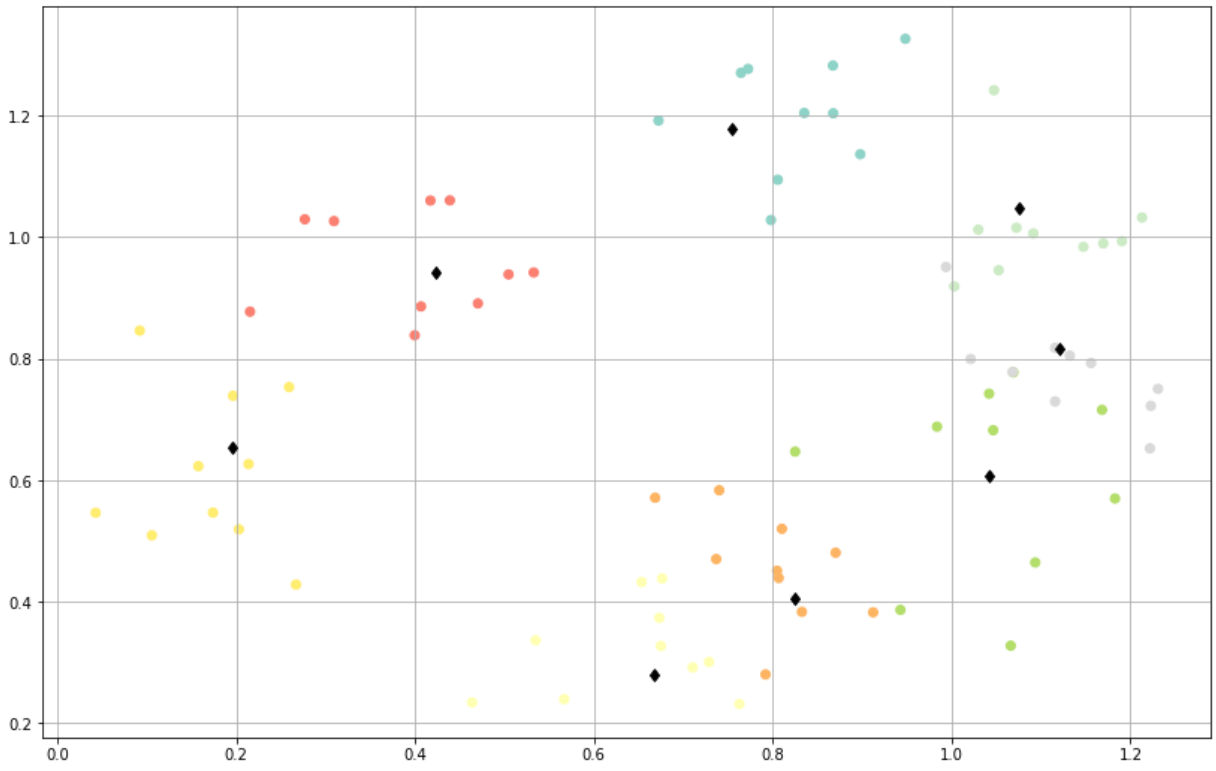
Задание №1

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

Формируем входные данные для сети Кохонена

```
x, y, centers = make_blobs(
    n_samples=80, cluster_std=0.1, n_features=2,
    centers=8, center_box=(0, 1.5),
    return_centers=True, random_state=177
)
```

```
plt.figure(figsize=(14, 9))
plt.scatter(x[:, 0], x[:, 1], c=y, cmap='Set3')
plt.plot(centers[:,0], centers[:, 1], 'dk')
plt.grid(True)
```



Построим и обучим слой Кохонена

```
clusters = 8
competNet = nl.net.newc([[0.0, 0.8],[0.0, 1.3]], clusters)
error = competNet.train(x, epochs=500, show=100)

plt.figure(figsize=(14, 9))
plt.title('Task №1')
plt.plot(error)
plt.grid()
plt.xlabel('Epoch number')
plt.ylabel('Error (default MAE)')
None
```

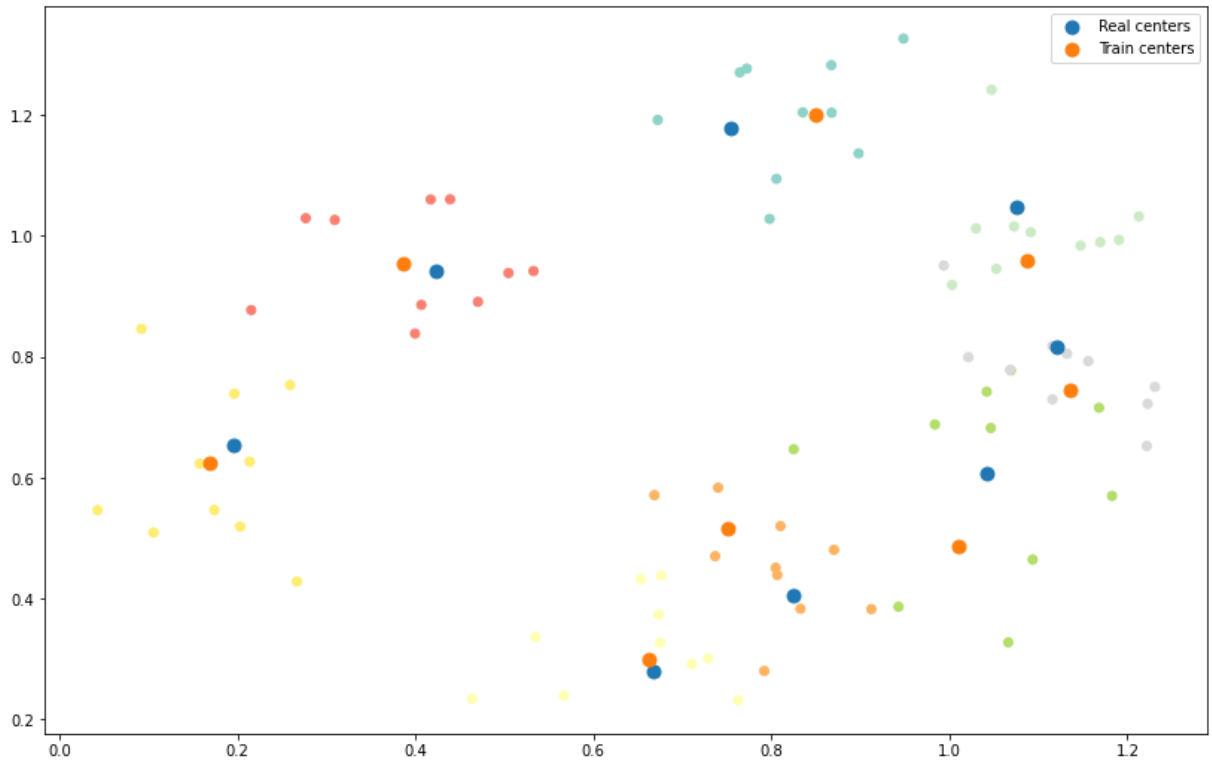
```
Epoch: 100; Error: 13.588213968980277;
Epoch: 200; Error: 13.096939627836043;
Epoch: 300; Error: 11.346044138391775;
Epoch: 400; Error: 10.831157697644903;
Epoch: 500; Error: 10.86257702812902;
The maximum number of train epochs is reached
```

Веса (центры кластеров) полученной сети:

```
weightsAreCenter = competNet.layers[0].np['w']

plt.figure(figsize=(14, 9))
plt.scatter(x[:, 0], x[:, 1], c=y, cmap='Set3')
plt.scatter(centers[:,0], centers[:, 1], s = 80, label="Real centers")
plt.scatter(weightsAreCenter[:,0], weightsAreCenter[:,1], s = 80, label="Train centers")
plt.legend()
plt.show()

weightsAreCenter
```



Создадим случайным образом 5 точек классифицируем их:

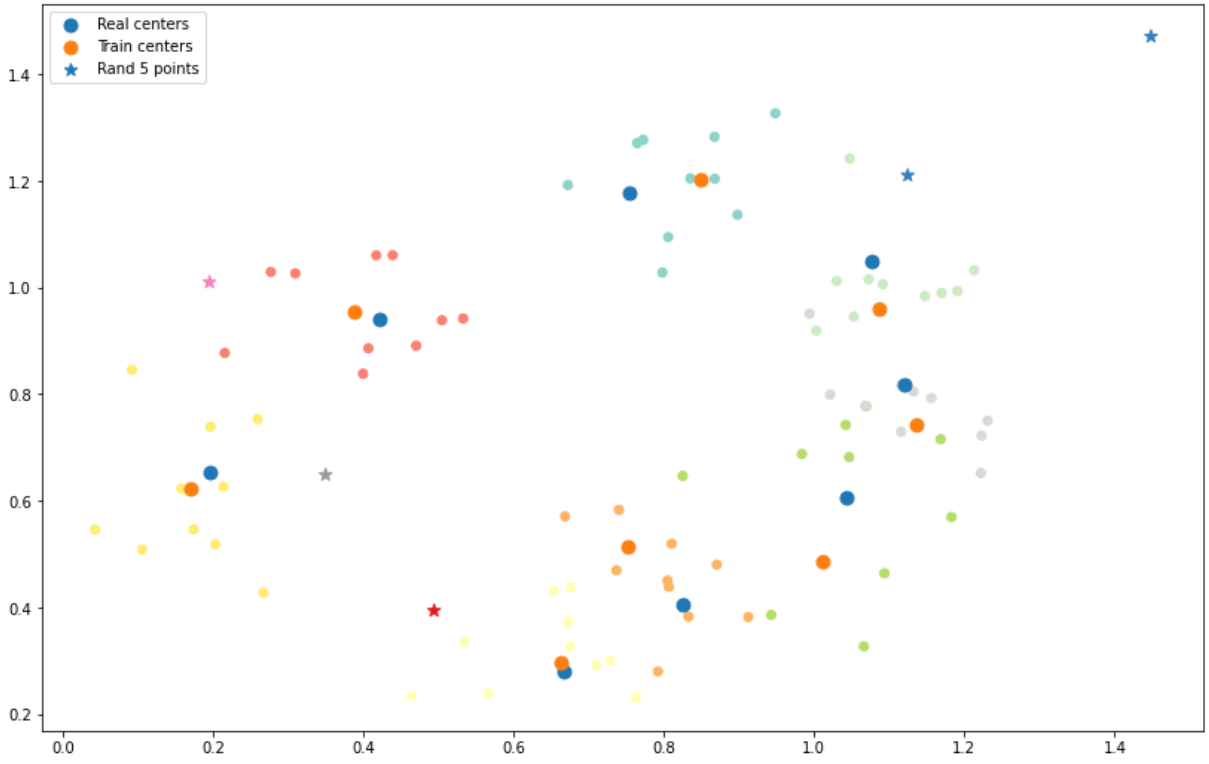
```
randPoints = np.array([[np.random.uniform(0, 1.5),
                        np.random.uniform(0, 1.5)] for _ in range(5)])

pred = competNet.sim(randPoints)
classPred = np.argmax(pred, axis=1)

print("Классы:", classPred)

plt.figure(figsize=(14, 9))
plt.scatter(x[:, 0], x[:, 1], c=y, cmap='Set3')
plt.scatter(centers[:,0], centers[:, 1], s = 80, label="Real centers")
plt.scatter(weightsAreCenter[:,0], weightsAreCenter[:,1], s = 80, label="Train centers")
plt.scatter(randPoints[:, 0], randPoints[:, 1], c=classPred,
            cmap='Set1', marker='*', s=80, label='Rand 5 points')
plt.legend()
plt.show()
```

```
Классы: [5 4 1 1 0]
```



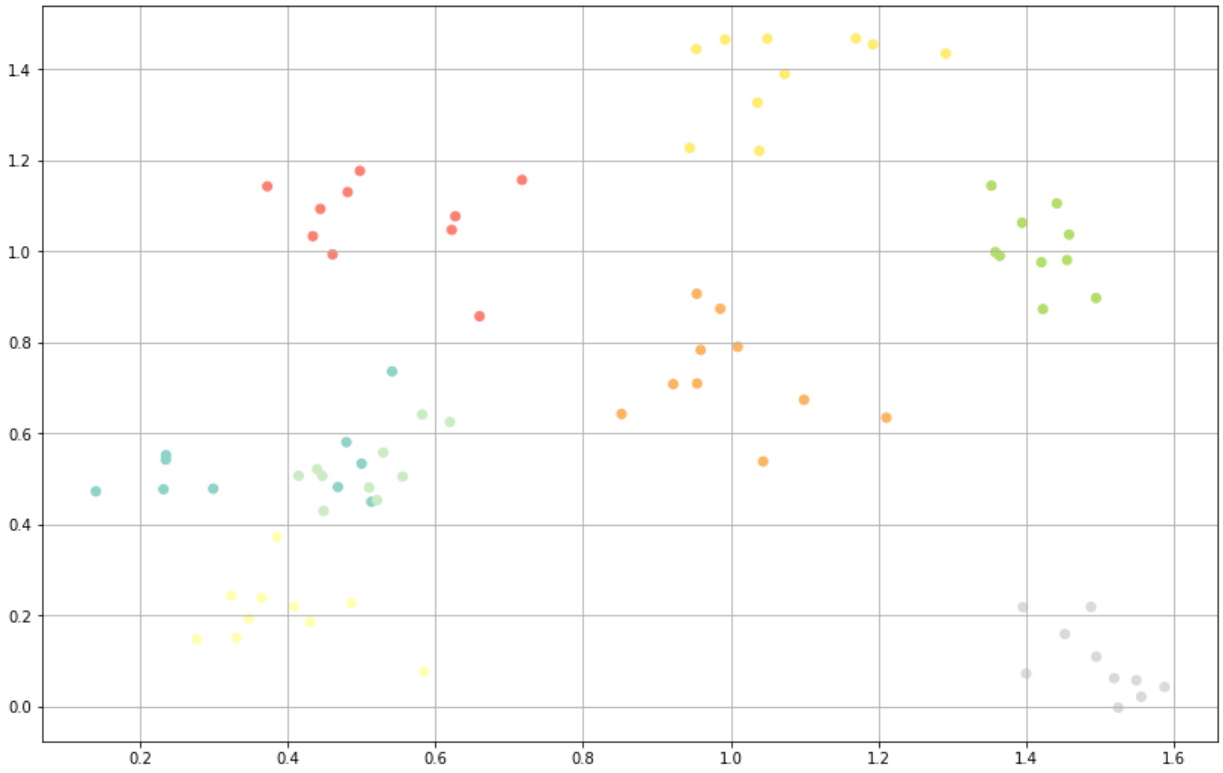
Задание №2

Нужно построить и обучить карту Кохонена размера 2x4 с гексагональной сеткой

Сгенерируем набор из точек:

```
x2, y2 = make_blobs(n_samples=80, cluster_std=0.1, n_features=2,
                    centers=8, center_box=(0, 1.5), random_state=91)

plt.figure(figsize=(14, 9))
plt.scatter(x2[:, 0], x2[:, 1], c=y2, cmap='Set3')
plt.grid(True)
```



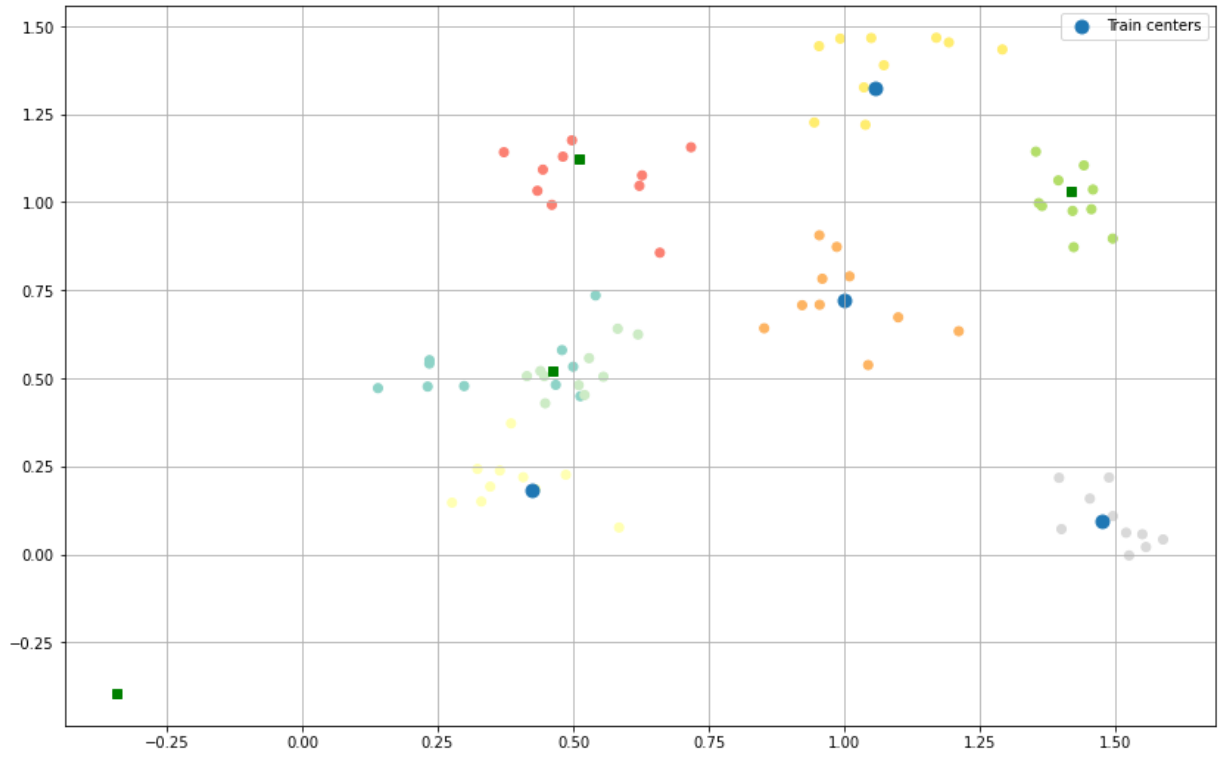
Обучим сеть:

```
epochs = 150
som = MiniSom(2, 4, x2.shape[1], sigma=0.66, learning_rate=0.8, activation_distance='euclidean',
              topology='hexagonal', neighborhood_function='gaussian', random_seed=10)
som.train(x2, epochs, verbose=True)
```

```
 [ 150 / 150 ] 100% - 0:00:00 left
 quantization error: 0.11702417615350655
```

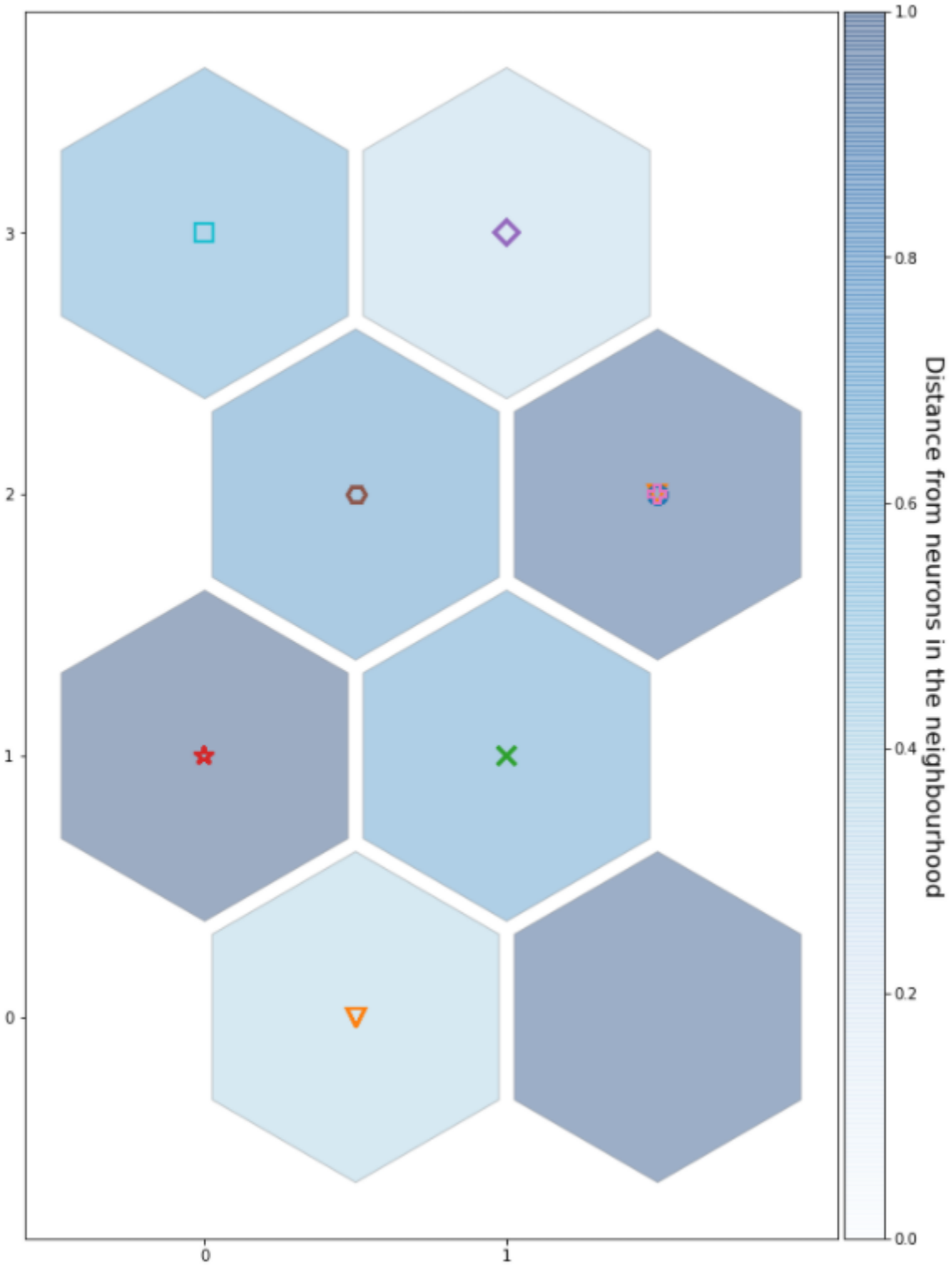
```
xx, yy = som.get_euclidean_coordinates()
umatrix = som.distance_map()
weights = som.get_weights()
```

```
plt.figure(figsize=(14, 9))
plt.scatter(x2[:, 0], x2[:, 1], c=y2, cmap='Set3')
plt.scatter(weights[0][:,0], weights[0][:,1], s=80, label='Train centers')
plt.plot(weights[1][:,0], weights[1][:,1], 'gs')
plt.legend()
plt.grid(True)
```



Гексогональная сетка для обучающего множества.

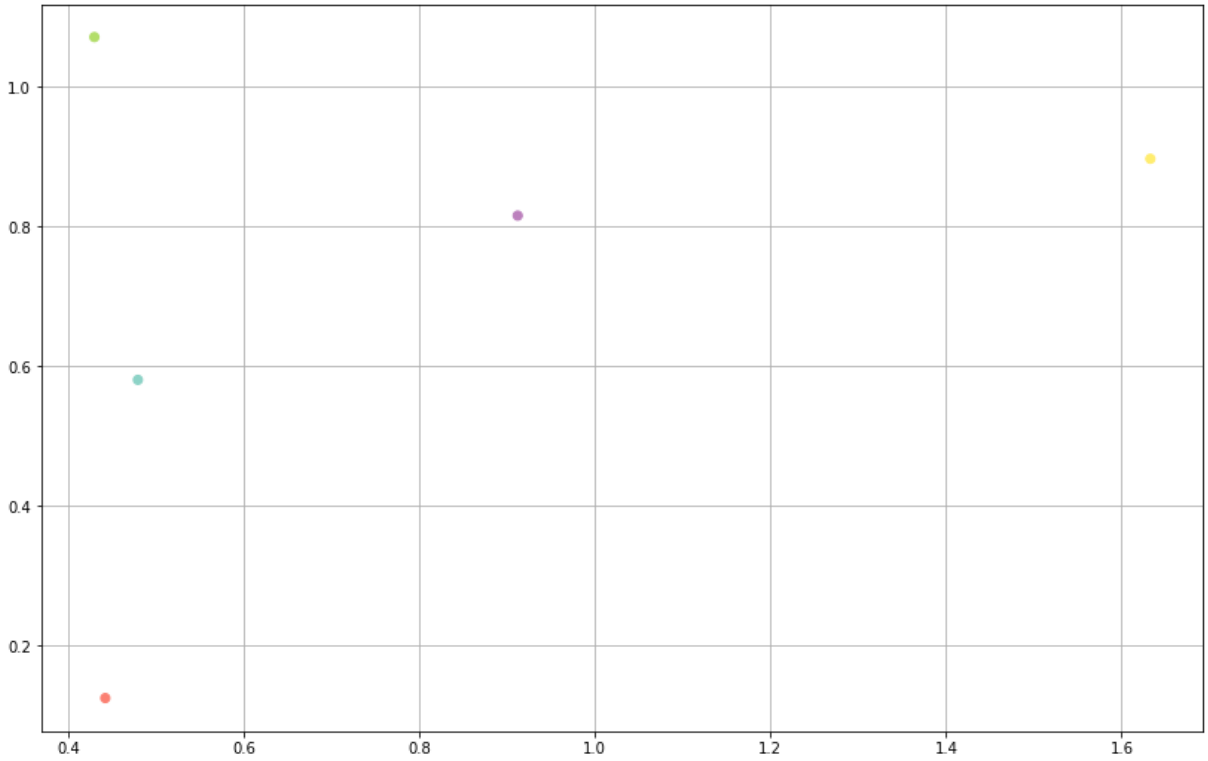
```
DrawHex(x2, y2, xx, yy, umatrix, weights)
```

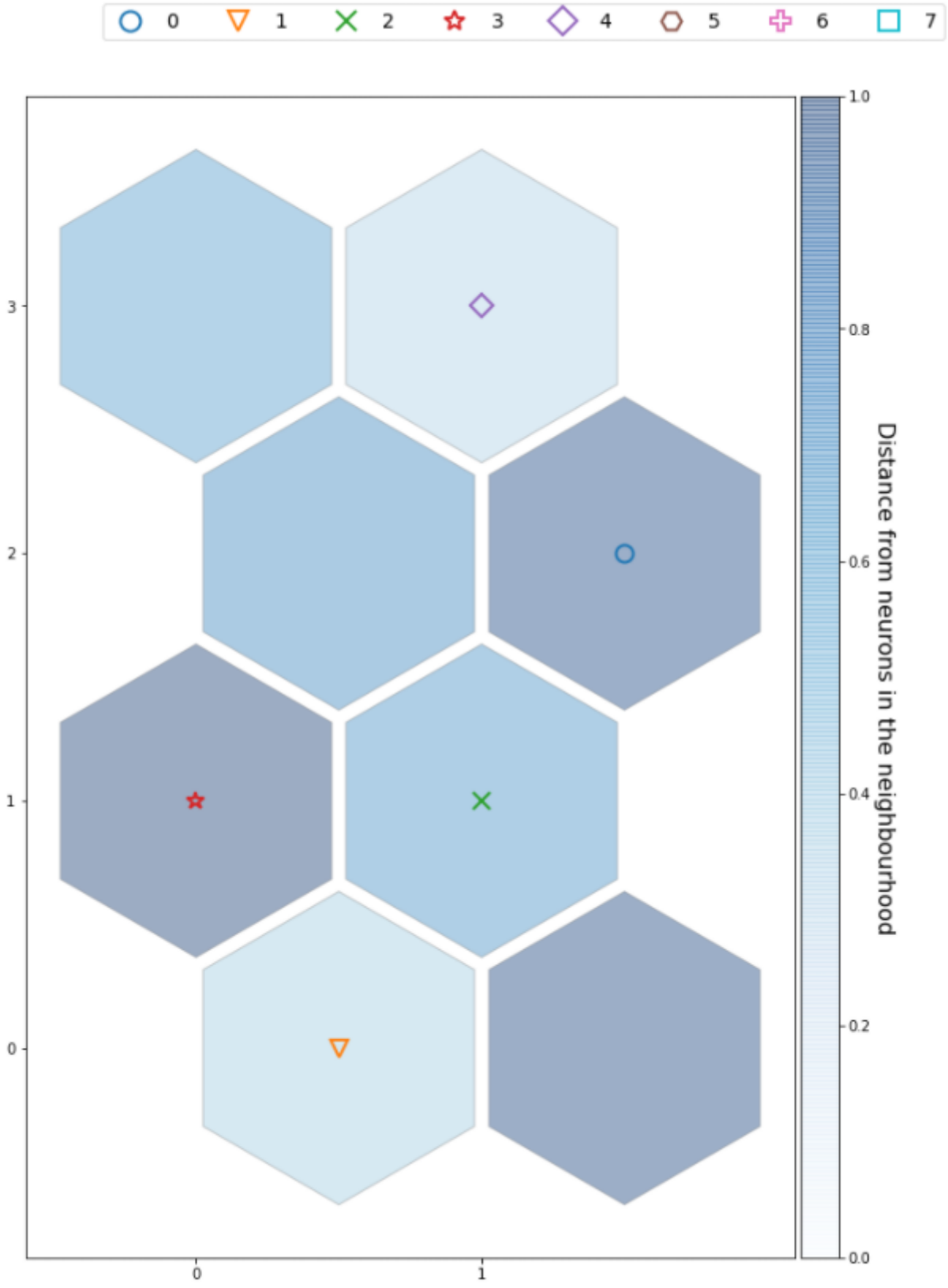

3
2
1
0
0
1
1.0
0.8
0.6
0.4
0.2
0.0
Distance from neurons in the neighbourhood

Сгенерируем случайные точки и определим их кластеры:

```
randx, randy = make_blobs(n_samples=5, cluster_std=0.1, n_features=2,
                    centers=8, center_box=(0, 1.5), random_state=91)

plt.figure(figsize=(14, 9))
plt.scatter(randx[:, 0], randx[:, 1], c=randy, cmap='Set3')
plt.grid(True)
```




0
1
2
3
4
5
6
7
3
2
1
0
0
1
1.0
0.8
0.6
0.4
0.2
0.0
Distance from neurons in the neighbourhood

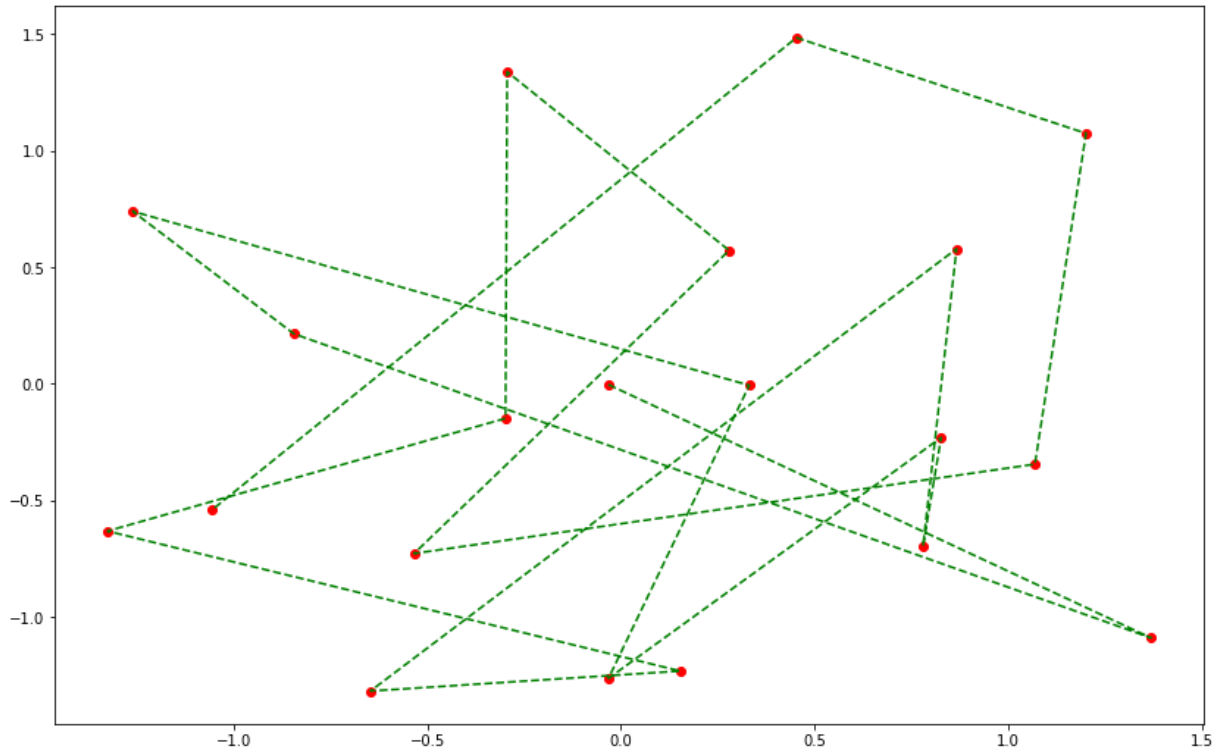
Задание №3

Использовать карту Кохонена для нахождения одного из решений задачи коммивояжера.

Сгенерируем набор из 20 случайных точек из диапазона $[-1.5, 1.5]$.

```
z = np.array([[np.random.uniform(-1.5, 1.5), np.random.uniform(-1.5, 1.5)] for _ in range(20)])

plt.figure(figsize=(14, 9))
plt.plot(z[:, 0], z[:, 1], '--', c='green')
plt.scatter(z[:, 0], z[:, 1], c='red');
```



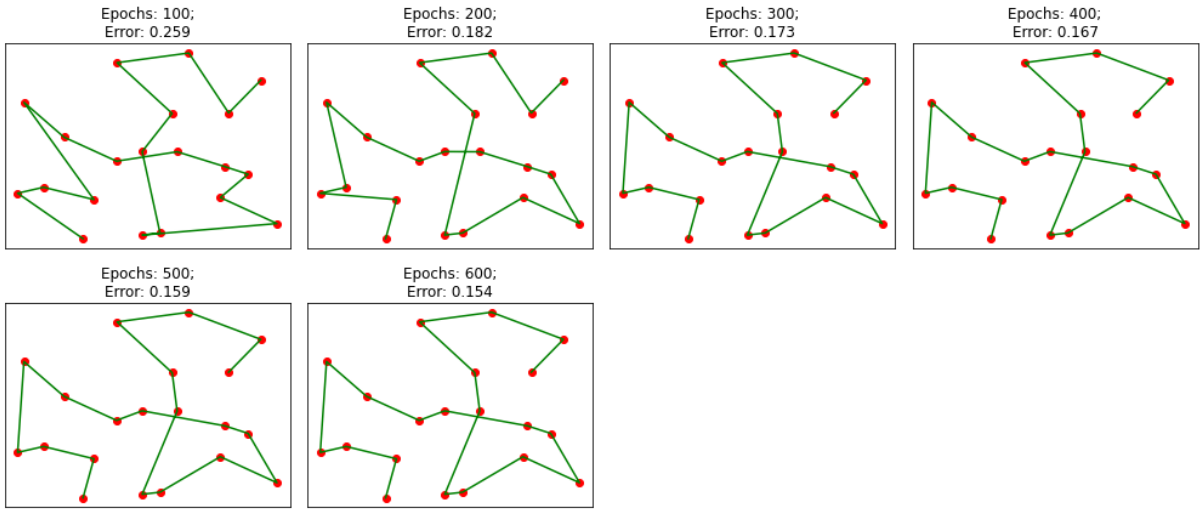
Обучим сеть и посмотрим на полученный результат:

```
np.random.RandomState(10)
neurons = 80

som = MiniSom(1, neurons, z.shape[1], sigma=8, learning_rate=0.4,
              neighborhood_function='gaussian', random_seed=0)
som.random_weights_init(z)

plt.figure(figsize=(14, 9))
for i, iterations in enumerate(range(100, 601, 100)):
    som.train(z, iterations, verbose=False, random_order=False)
    plt.subplot(3, 4, i+1)
    plt.scatter(z[:, 0], z[:, 1], c='red')
    visit_order = np.argsort([som.winner(p)[1] for p in z])
    #visit_order = np.concatenate((visit_order, [visit_order[0]]))
    plt.plot(z[visit_order][:,0], z[visit_order][:,1], c='green')
    plt.title("Epochs: {i};\nError: {e:.3f}".format(i=iterations,
                                                    e=som.quantization_error(z)))
    plt.xticks([])
    plt.yticks([])
plt.tight_layout()
plt.show()
```

```
/home/artoy/.local/lib/python3.6/site-packages/minisom.py:154: UserWarning: Warning: sigma is too high for the dimension of the map.
  warn('Warning: sigma is too high for the dimension of the map.')
```



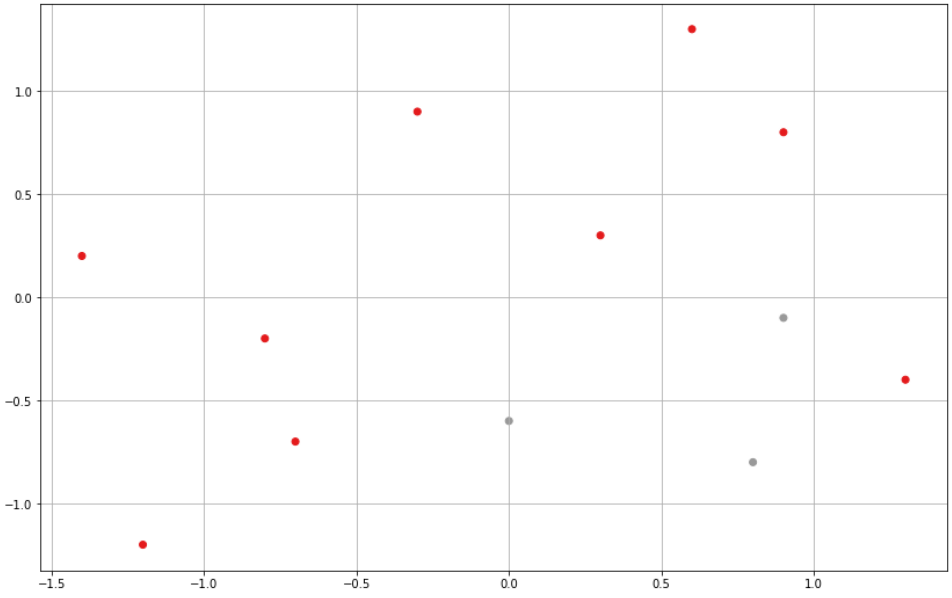
Задание №4

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

```
points = np.array([
    [0.9, 0.9, 0.6, 0, 0.8, -0.8, 0.3, -1.2, -0.7, -0.3, 1.3, -1.4],
    [0.8, -0.1, 1.3, -0.6, -0.8, -0.2, 0.3, -1.2, -0.7, 0.9, -0.4, 0.2]
])

target = np.array([0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0])
pointsT = points.T

plt.figure(figsize=(14, 9))
plt.scatter(pointsT[:, 0], pointsT[:, 1], c=target, cmap='Set1')
plt.grid(True)
```



Строим LVQ-сеть и обучаем на 300 эпохах.

```
lvqnet = LVQ(n_inputs=2, n_classes=2, step=0.1)
lvqnet.train(pointsT, target, epochs=300)
```

Классифицируем точки области $[-1.5; 1.5]x[-1.5; 1.5]$ с шагом 0.1.

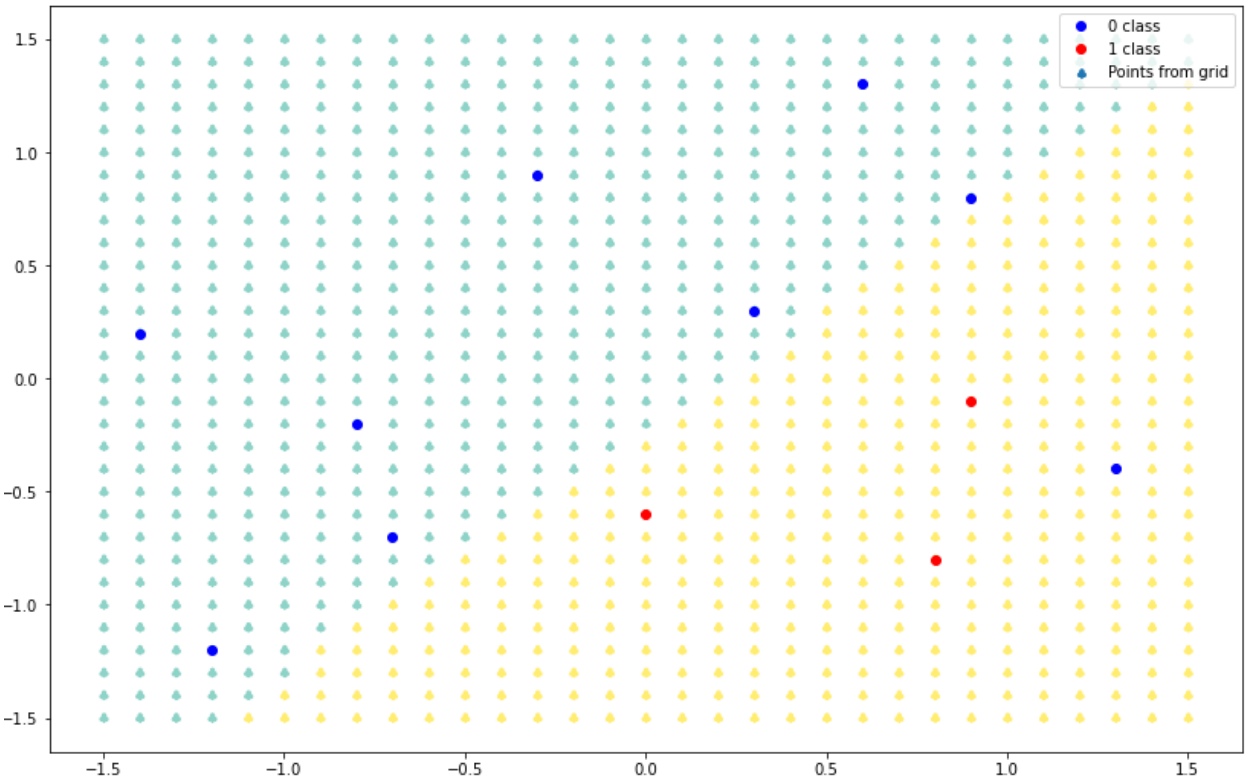
```
xx, yy = np.meshgrid(np.arange(-1.5, 1.51, 0.1), np.arange(-1.5, 1.51, 0.1))
xx.shape = xx.size, 1
yy.shape = yy.size, 1
sample = np.concatenate((xx, yy), axis=1)

pred = lvqnet.predict(sample)
```

Полученная сетка:

```
plt.figure(figsize=(14, 9))

plt.scatter(sample[:, 0], sample[:, 1], c=pred,
            cmap='Set3', marker=r'$\clubsuit$', label='Points from grid')
plt.plot([pointsT[i][0] for i in range(12) if target[i] == 0],
         [pointsT[i][1] for i in range(12) if target[i] == 0], 'bo', label='0 class')
plt.plot([pointsT[i][0] for i in range(12) if target[i] == 1],
         [pointsT[i][1] for i in range(12) if target[i] == 1], 'ro', label='1 class');
plt.legend()
plt.show()
```



ВЫВОДЫ

Выполнив шестую лабораторную работу по курсу Нейроинформатика, я узнал о сетях Кохонена, которые удивительно хорошо справляются с задачами кластеризации понижения размера данных.

Я рассмотрел несколько вариантов этих нейронных сетей, таких как LVQ-сети (сеть векторного квантования, обучаемая с учителем), SOM-сети (карта Кохонена) и простой слой Кохонена и попробовал применить их в различных задачах, тесно связанных с кластеризацией.