

**Липецкий государственный технический университет**

**Факультет автоматизации и информатики**

**Кафедра автоматизированных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**по дисциплине «Прикладные интеллектуальные системы и экспертные  
системы»**

**«Нейронные сети. Обучение без учителя»**

Студент

Косенков В.Д.

Группы М-ИАП-23

Руководитель

Кургасов В.В.

Доцент

Липецк 2023 г

## Цель работы

Использование нейронной сети Кохонена с самообучением для задачи кластеризации случайных точек на плоскости вокруг двух центров кластеризации.

### Задание кафедры

Применить нейронную сеть Кохонена с самообучением для задачи кластеризации. На первом этапе сгенерировать случайные точки на плоскости вокруг 2 центров кластеризации (примерно по 20-30 точек). Далее считать, что сеть имеет два входа (координаты точек) и два выхода – один из них равен 1, другой 0 (по тому, к какому кластеру принадлежит точка). Подавая последовательно на вход (вразнобой) точки, настроить сеть путем применения описанной процедуры обучения так, чтобы она приобрела способность определять, к какому кластеру принадлежит точка. Коэффициент выбрать, уменьшая его от шага к шагу по правилу  $a = (50-i)/100$ , причем для каждого нейрона это будет своё значение  $a$ , а подстраиваться на каждом шаге будут веса только одного (выигравшего) нейрона.

### Ход работы

Сгенерируем случайные точки вокруг двух центров кластеризации. Это представлено на рисунке 1.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs

# Создаем случайные точки вокруг двух центров кластеризации
data, labels = make_blobs(n_samples=60, centers=2, random_state=42)

# Построение графика сгенерированных точек
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', edgecolors='k')
plt.title('Generated Data')
plt.show()
```

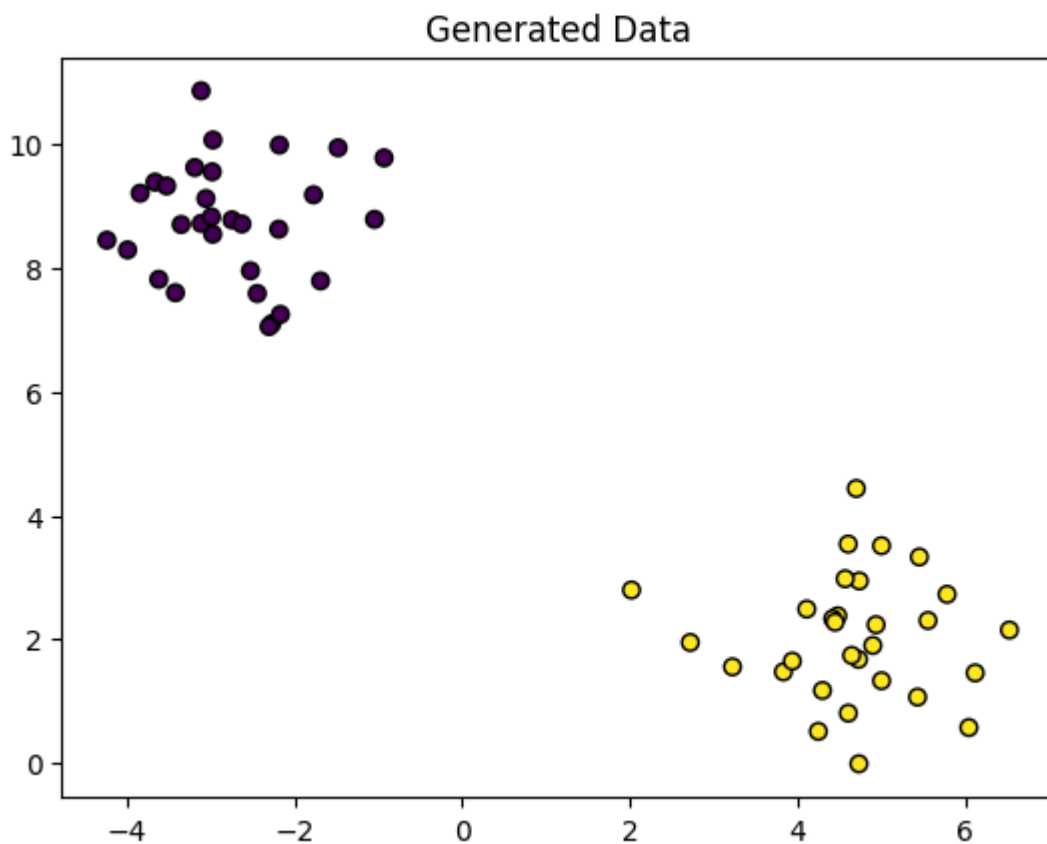


Рисунок 1 - Сгенерированные точки

Далее зададим параметры нейронной сети Кохонена, проинициализируем веса и запустим обучение, код для обучения, график первой эпохи представлен на рисунке 2.

```

# Инициализируем веса и коэффициент обучения
weights = np.random.rand(2, 2)
learning_rate = 0.5

# Функция обучения нейронной сети Кохонена
def train_kohonen_network(data, weights, learning_rate, epochs=50):
    for epoch in range(epochs):
        for point in data:
            # Рассчитываем расстояния между точкой и весами каждого нейрона
            distances = np.linalg.norm(point - weights, axis=1)

            # Находим индекс нейрона, который ближе всего к точке
            winner_index = np.argmin(distances)

            # Обновляем веса выбранного нейрона
            weights[winner_index] += learning_rate * (point - weights[winner_index])

        # Уменьшаем коэффициент обучения
        learning_rate = (50 - epoch) / 100

        # Построение карты сети Кохонена после первой эпохи обучения
        plt.figure(figsize=(8, 8))
        plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', edgecolors='k')
        plt.scatter(weights[:, 0], weights[:, 1], marker='X', s=200, c='r', label='Centroids')
        plt.title(f'Kohonen Network Clustering - Epoch {epoch + 1}')
        plt.legend()
        plt.show()

    return weights

# Обучаем нейронную сеть Кохонена
trained_weights = train_kohonen_network(data, weights, learning_rate)

# Построение итогового графика
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', edgecolors='k')
plt.scatter(trained_weights[:, 0], trained_weights[:, 1], marker='X', s=200, c='r', label='Centroids')
plt.title('Kohonen Network Clustering')
plt.legend()
plt.show()

```

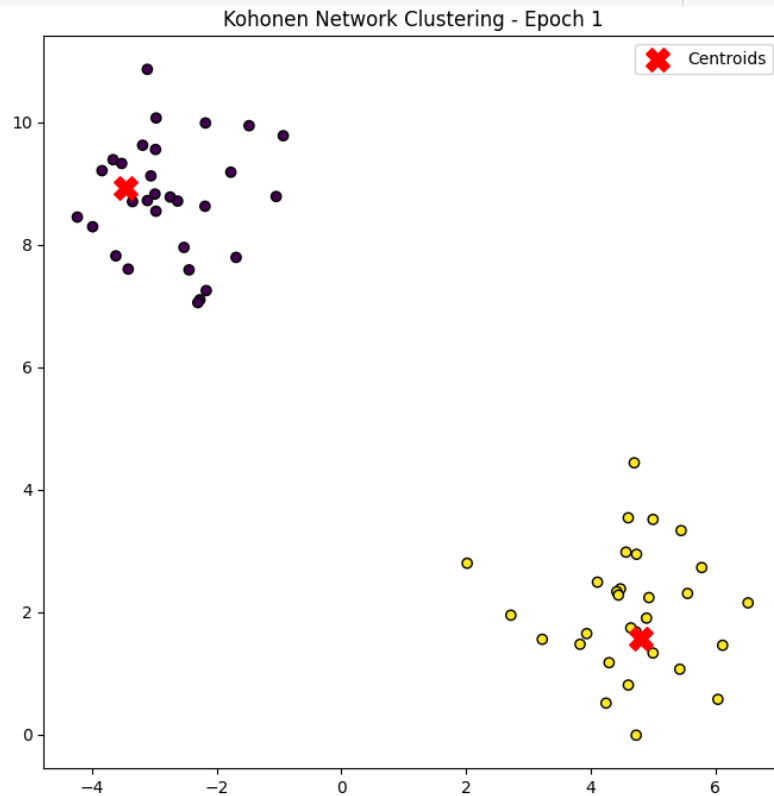


Рисунок 2 - Код и визуализация первой эпохи

Визуализация последней эпохи представлена на рисунке 3.

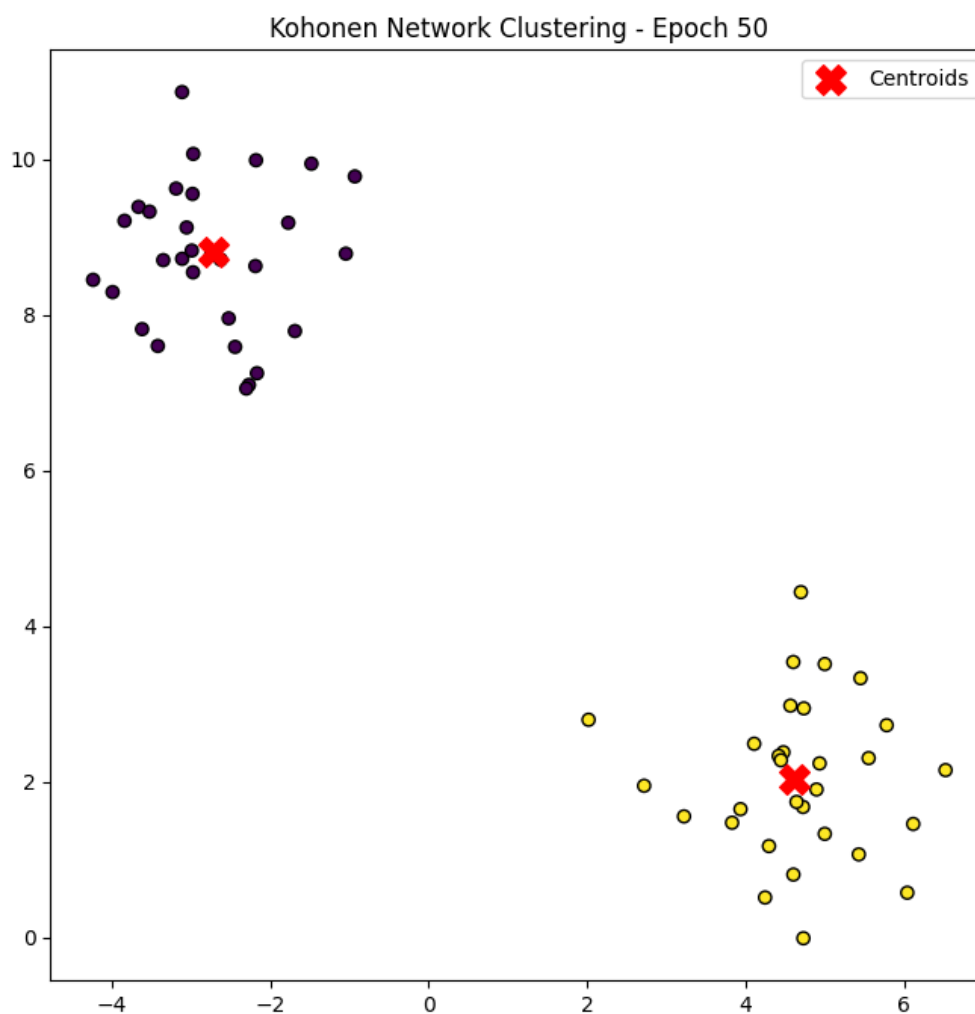


Рисунок 3 - Эпоха 50

После обучения можно использовать сеть для классификации точек, это показано на рисунке 4.

```
# Обучаем нейронную сеть Кохонена
trained_weights = train_kohonen_network(data, weights, learning_rate)

# Построение итогового графика
plt.scatter(data[:, 0], data[:, 1], c=labels, cmap='viridis', edgecolors='k')
plt.scatter(trained_weights[:, 0], trained_weights[:, 1], marker='X', s=200, c='r', label='Centroids')
plt.title('Kohonen Network Clustering')
plt.legend()
plt.show()
```

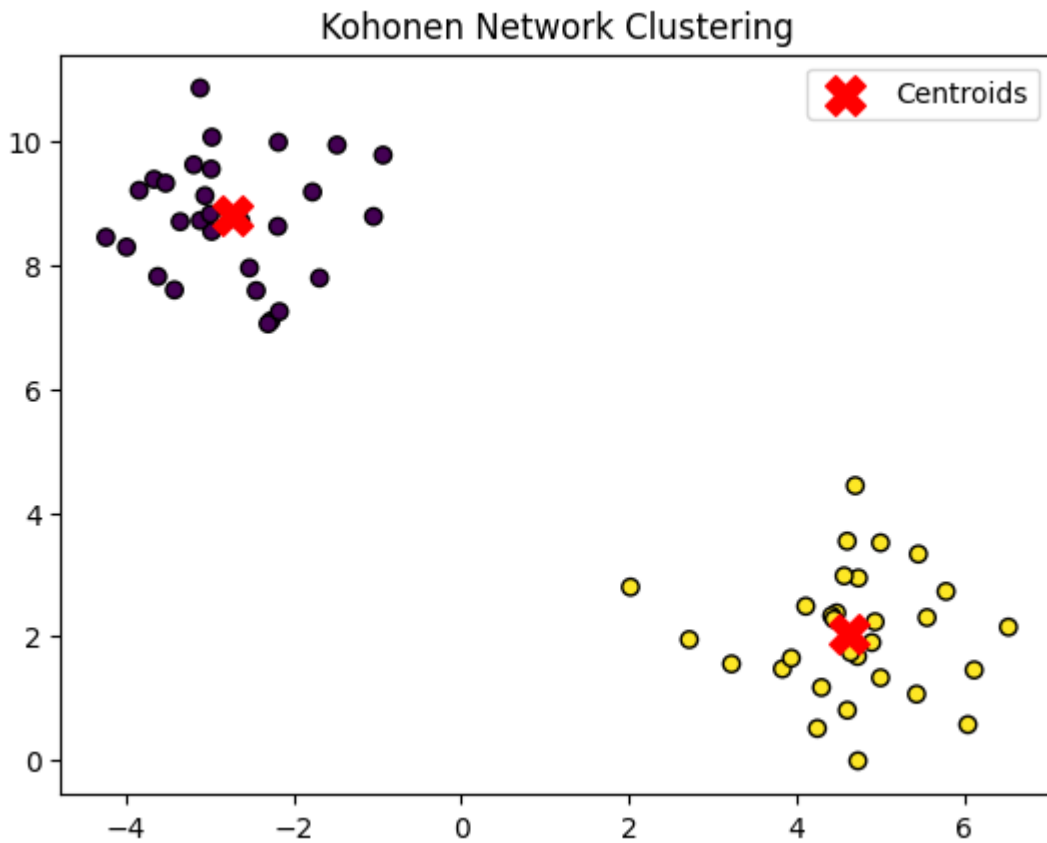


Рисунок 4 - Визуализация результатов

## Вывод

В результате выполнения работы были получены практические навыки использования нейронной сети Кохонена с самообучением для задачи кластеризации случайных точек на плоскости вокруг двух центров кластеризации.