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 Бахмутский М.В.

Группа М-ИАП-22

Руководитель Кургасов В.В.

Липецк 2022 г.

Задание кафедры

Применить нейронную сеть Кохонена с самообучение 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 α выбрать, уменьшая его от шага к шагу по правилу $\alpha = 50 - i100$, причем для каждого нейрона это будет свое значение α, а подстраиваться на каждом шаге будут веса только одного (выигравшего) нейрона.

Ход работы

1) Сгенерируем выборку с помощью функции make_blobs. Данная операция представлена на рисунке 1.

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
import numpy as np
from scipy.cluster.hierarchy import fcluster, linkage
import math
from sklearn.metrics import accuracy_score
```

Рисунок 1 – Сгенерированная выборка

2) Выделим два кластера и обозначим их центры, полученный график представлен на рисунке 2.

```
plt.scatter(X[:, 0], X[:, 1], c=T)
plt.scatter(clusters[:, 0], clusters[:, 1], c='blue')
```

```
<matplotlib.collections.PathCollection at 0x7efbd88b9850>
```

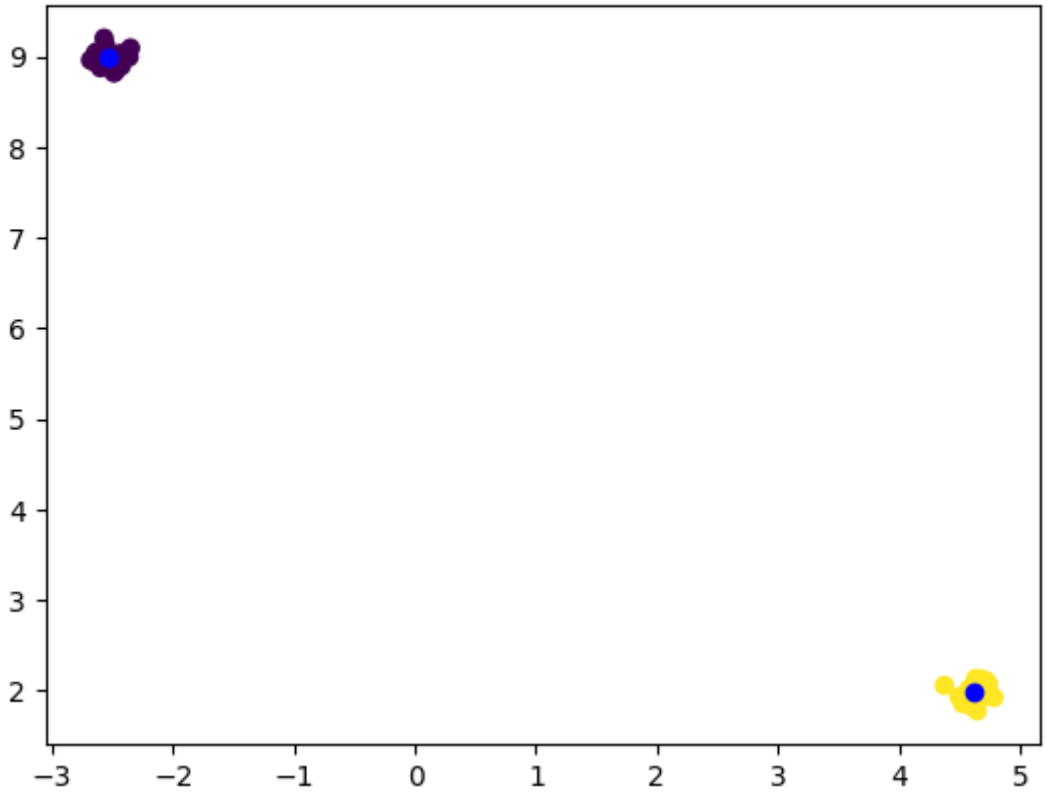

Рисунок 2 – Выделение кластеров

3) Для работы нейросети Кохонена необходимо сгегерировать веса, которые представлены на рисунке 3.

```
mergings = linkage(X, method='ward')
T = fcluster(mergings, 2, criterion='maxclust')
clusters = update_cluster_centers(X, T)
clusters
```

```
array([[-2.53316572,  8.9825378 ],
       [ 4.62510186,  1.98008378]])
```

Рисунок 3 – Веса нейросети

4) Последовательное обновление весов представлено на рисунке 4;

```
def update_cluster_centers(X, c):
    centers = np.zeros((2, 2))
    for i in range(1, 3):
        ix = np.where(c == i)
        centers[i - 1, :] = np.mean(X[ix, :], axis=1)
    return centers
```

Рисунок 4 – Обновление весов

5) Итоговые веса представлены на рисунке 5:

```
s = X[0]
J = som.winner(weights, s)

print(f"Элемент принадлежит к {J} кластеру, на самом деле к {y[0]} кластеру"
print("Обученные веса: ")
print(weights)
```

```
Элемент принадлежит к 1 кластеру, на самом деле к 1 кластеру
Обученные веса:
[[3.33757957 3.19914652]
 [2.18725367 4.34796543]]
```

Рисунок 5 – Итоговые веса

6) Итоговое качество кластеризации представлено на рисунке 6

```
y == predicted
```

```
array([ True,  True, False, False, False,  True, False,  True,  True,
        True, False, False, False, False,  True,  True,  True, False,
       False,  True, False,  True,  True,  True,  True, False,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
       False, False,  True, False, False,  True,  True,  True,  True,
        True,  True,  True,  True, False])
```

```
print(f'Точность кластеризации: {accuracy_score(y, predicted) * 100}%')
```

```
Точность кластеризации: 64.0%
```

Рисунок 6 – Точность классификации

Вывод

В ходе выполнения данной лабораторной работы мною были получены навыки построения нейронной сети Кохонена с самообучения для решения задачи кластеризации. После успешного построения и обучения модели была рассчитана характеристика точности классификации точек к их кластерам.