**Липецкий государственный технический университет**

Факультет автоматизации и информатики

Кафедра автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №6

по дисциплине

«Прикладные интеллектуальные системы и экспертные системы»

Студент Первушин О. С.

Группа М-ИАП-23-1

Руководитель Кургасов В. В.

доцент, канд. пед. наук

Липецк 2023 г.

Цель работы

Применить нейронную сеть Кохонена с самообучением для задачи 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 = (50-i)/100, причем для каждого нейрона это будет свое значение, а подстраиваться на каждом шаге будут веса только одного (выигравшего) нейрона.

Ход работы

Реализуем сеть Кохонена на языке Python с заданными критериями (листинг 1), а также приведем пример отчета, выданного программой (листинг 2).

Листинг 1 — Код программы

import numpy as np  
  
np.random.seed(0)  
  
# Генерация случайных точек вокруг центров кластеризации  
cluster1\_center = [2, 2]  
cluster1\_points = np.random.randn(20, 2) + cluster1\_center  
  
cluster2\_center = [-2, -2]  
cluster2\_points = np.random.randn(20, 2) + cluster2\_center  
  
# Координаты всех точек  
points = np.concatenate([cluster1\_points, cluster2\_points])  
  
# Создание сети Кохонена с заданными входами и количеством нейронов  
num\_inputs = 2  
num\_neurons = 2  
weights = np.random.randn(num\_neurons, num\_inputs)  
  
# Обучение сети путем последовательного предъявления точек  
learning\_rate = 0.01  
num\_epochs = 50  
  
for epoch in range(num\_epochs):  
 for point in points:  
 # Распространение сигнала через сеть  
 distances = np.linalg.norm(point – weights, axis=1)  
 winner\_neuron = np.argmin(distances)  
  
 # Обновление весов выигравшего нейрона  
 weights[winner\_neuron] += learning\_rate \* (point – weights[winner\_neuron])  
  
 # Уменьшение коэффициента выбора  
 learning\_rate = (50 – epoch) / 100  
  
# Проверка принадлежности точек к кластерам  
for point in points:  
 distances = np.linalg.norm(point – weights, axis=1)  
 winner\_neuron = np.argmin(distances)  
 cluster\_label = «Cluster 1» if winner\_neuron == 0 else «Cluster 2»  
 print(f»Point {point} belongs to {cluster\_label}»)

Листинг 2 — Вывод программы (консоль)

Point [3.76405235 2.40015721] belongs to Cluster 1

Point [2.97873798 4.2408932 ] belongs to Cluster 1

Point [3.86755799 1.02272212] belongs to Cluster 1

Point [2.95008842 1.84864279] belongs to Cluster 1

Point [1.89678115 2.4105985 ] belongs to Cluster 1

Point [2.14404357 3.45427351] belongs to Cluster 1

Point [2.76103773 2.12167502] belongs to Cluster 1

Point [2.44386323 2.33367433] belongs to Cluster 1

Point [3.49407907 1.79484174] belongs to Cluster 1

Point [2.3130677 1.14590426] belongs to Cluster 1

Point [-0.55298982 2.6536186 ] belongs to Cluster 1

Point [2.8644362 1.25783498] belongs to Cluster 1

Point [4.26975462 0.54563433] belongs to Cluster 1

Point [2.04575852 1.81281615] belongs to Cluster 1

Point [3.53277921 3.46935877] belongs to Cluster 1

Point [2.15494743 2.37816252] belongs to Cluster 1

Point [1.11221425 0.01920353] belongs to Cluster 1

Point [1.65208785 2.15634897] belongs to Cluster 1

Point [3.23029068 3.20237985] belongs to Cluster 1

Point [1.61267318 1.69769725] belongs to Cluster 1

Point [-3.04855297 -3.42001794] belongs to Cluster 2

Point [-3.70627019 -0.0492246 ] belongs to Cluster 2

Point [-2.50965218 -2.4380743 ] belongs to Cluster 2

Point [-3.25279536 -1.22250964] belongs to Cluster 2

Point [-3.61389785 -2.21274028] belongs to Cluster 2

Point [-2.89546656 -1.6130975 ] belongs to Cluster 2

Point [-2.51080514 -3.18063218] belongs to Cluster 2

Point [-2.02818223 -1.57166813] belongs to Cluster 2

Point [-1.93348278 -1.6975281 ] belongs to Cluster 2

Point [-2.63432209 -2.36274117] belongs to Cluster 2

Point [-2.67246045 -2.35955316] belongs to Cluster 2

Point [-2.81314628 -3.7262826 ] belongs to Cluster 2

Point [-1.82257386 -2.40178094] belongs to Cluster 2

Point [-3.63019835 -1.53721774] belongs to Cluster 2

Point [-2.90729836 -1.9480546 ] belongs to Cluster 2

Point [-1.27090944 -1.87101709] belongs to Cluster 2

Point [-0.86059932 -3.23482582] belongs to Cluster 2

Point [-1.59765836 -2.68481009] belongs to Cluster 2

Point [-2.87079715 -2.57884966] belongs to Cluster 2

Point [-2.31155253 -1.94383466] belongs to Cluster 2

Вывод

Мы применили нейронную сеть Кохонена с самообучением для задачи кластеризации.