Министерство науки и высшего образованияРоссийской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по практическому занятию №4

на тему

«Кластерный анализ данных»

по дисциплине

«**Машинное обучение**»

Выполнили:

Студенты группы №140

Бригада 7

Сафаров Д. А.

Тимохин Е. М.

Проверила:

ас. Панина И.С.

**Цель работы**

Изучить простейший метод кластерного анализа данных.

**Задание**

Построить нейронную сеть, производящую кластеризацию методом k-means, создав по 600 объектов на облако, в соответствии с вариантом: 3 кластера, 5 облаков.

**Практическая часть**

import numpy as np

import matplotlib.pyplot as plt

# Функция стоимости для алгоритма k-средних, которая измеряет,

# насколько хорошо центры кластеров соответствуют данным

# и представляет собой сумму квадратов расстояний между точками данных и центрами их кластеров.

def cost(X, R, M):

cost = 0

for k in range(len(M)):

diff = X - M[k]

sq\_distances = (diff \* diff).sum(axis=1)

cost += (R[:, k] \* sq\_distances).sum()

return cost

def plot\_k\_means(X, K, max\_iter=20, beta=1.0):

N, D = X.shape

M = np.zeros((K, D))

R = np.zeros((N, K))

for k in range(K):

M[k] = X[np.random.choice(N)]

costs = []

for i in range(max\_iter):

for k in range(K):

for n in range(N):

#вычисления расстояния для каждой точки данных X[n] и для каждого

центра кластера M[j]

R[n, k] = np.exp(-beta \* d(M[k], X[n])) / np.sum(np.exp(-beta \* d(M[j], X[n])) for j in range(K))

# обновление центров кластеров - среднее значение точек, принадлежащее

конкретному кластеру, взвешеное вероятностями принадлежности

# (exp - используется для вычисления вероятности принадлежности точки к

кластеру на основе расстояний )

M[k] = R[:, k].dot(X) / R[:, k].sum()

c = cost(X, R, M)

costs.append(c)

if i > 0 and np.abs(costs[-1] - costs[-2]) < 1e-5:

break

random\_colors = np.random.random((K, 3))

colors = R.dot(random\_colors)

plt.scatter(X[:, 0], X[:, 1], c=colors)

plt.show()

return M, R

# функция для вычисления квадрата евклидова расстояния между двумя точками

def d(u, v):

diff = u - v

return diff.dot(diff)

'''

В данной реализации кластер = облако, хотя по хорошему должно быть на каждый кластер по облаку

'''

def main():

D = 2 # количество признаков

s = 5 #параметр для определения координат для центров облаков

mu1 = np.array([-1.5, -1.5]) #центр первого кластера

mu2 = np.array([s+5, s+5]) #центр второго кластера

mu3 = np.array([0, s+5]) #центр третьего кластера

mu4=np.array([s+5,-2])

mu5=np.array([s+1,s-1])

N = 3000 # общее количество точек данных

X = np.zeros((N, D)) # матрица, где каждая строка - точка данных с D признаками

X[:600, :] = np.random.randn(600, D) + mu1 # Первые 600 точек генерируются из

нормального распределения с центром mu1

X[600:1200, :] = np.random.randn(600, D) + mu2 # Следующие 600 точек

генерируются с центром в mu2

X[1200:1800, :] = np.random.randn(600, D) + mu3 # Последние 600 точек

генерируются с центром в mu3

X[1800:2400,:]=np.random.randn(600,D)+mu4

X[2400:,:]=np.random.randn(600,D)+mu5

plt.scatter(X[:, 0], X[:, 1])#график рассеяния для визуализации;

X[:, 0] - ось X, X[:, 1] - ось Y

plt.xlabel('Ось X')

plt.ylabel('Ось Y')

plt.show()

K = 3 #количество кластеров

M, R = plot\_k\_means(X, K) # возвращает M - центры кластеров и R - матрица

принадлежности точек к кластерам

if \_\_name\_\_ == "\_\_main\_\_":

main()

На рисунке 1 представлен результат работы программы.

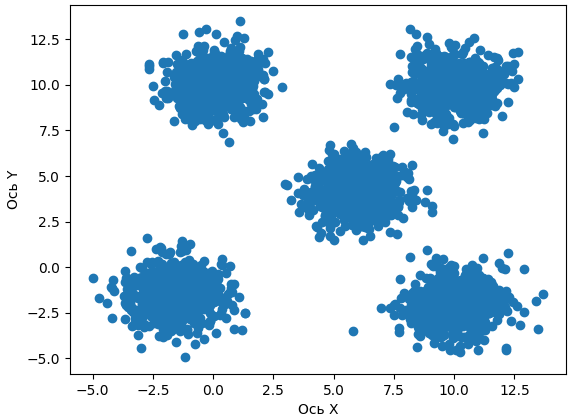


Рисунок 1 – Результат работы программы