|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение высшего образования**  **Московский государственный технический университет**  **имени Н.Э. Баумана** |

Факультет: Информатика и системы управления

Кафедра: Информационная безопасность

**Интеллектуальные технологии информационной безопасности**

**Лабораторная работа №6 на тему:**

«Алгоритмы кластерного анализа данных»

Вариант 4

Студент: Евула А. С.

Группа: ИУ8-63

Москва 2021

Цель работы:

Исследовать применение основных алгоритмов кластерного анализа, включая их модификации, на примере различных типов данных.

Постановка задачи:

Выполнить разбиение предложенных данных с помощью указанного в варианте алгоритма с использованием соответствующих метрик.

**Алгоритм**: НС Кохонена

**Исходные кластеризуемые данные**: Выборка лицеев г. Москвы. Координаты местоположения (X, Y).

**Расстояние**: Принадлежность округу Москвы (евклидово расстояние до координат (X0, Y0) центра округа).

Данные о лицеях и округах получены с сайта <https://data.mos.ru>

Ход работы

Сначала были созданы два файла с расширением .xlsx:

1. Список лицеев (название, координаты, округ)
2. Список округов (название, координаты)

Дальше для каждого лицея находится ближайший округ и этому нейрону присваивается значение “1”, остальным – ”0”, проводится коррекция весов.

Данные алгоритм проводится, пока ошибка не будет равна 0 (т.е. каждому лицею нейронная сеть не будет сопоставлять правильный округ).

По исходным данным, с нормой обучения равной 0.3, нейронная сеть обучилась за 2 эпохи.

Исходный код программы, ее вывод, список данных представлены в Приложении.

Приложение

Исходный код программы (НС)

# Copyright 2021, Evula A. S., All rights reserved.

# IC8-63 BMSTU

import pandas as pd

from random import random

from math import sqrt, inf

# liceum\_key = {'name': 0, 'X': 1, 'Y': 2, 'district': 3}

# district\_key = {'name': 0, 'X': 1, 'Y': 2}

class NN:

def \_\_init\_\_(self, file\_liceums, file\_districts):

self.liceums = pd.read\_excel(file\_liceums+'.xlsx', header=0, engine='openpyxl').values.tolist()

self.districts = pd.read\_excel(file\_districts+'.xlsx', header=0, engine='openpyxl').drop(columns=['Unnamed: 0']).values.tolist()

self.Rate = 0.3

self.W = []

for \_ in range(len(self.districts)):

temp = []

for \_ in range(len(self.liceums)):

temp.append([round(random(), 2), round(random(), 2)])

self.W.append(temp)

def train(self, debug=False):

epoch = -1

error = inf

while error > 0 and epoch < 1000:

epoch += 1

error = self.tick(debug)

print(f'\nK: {epoch}')

print(f'E: {error}/{len(self.liceums)}')

def tick(self, debug):

error = 0

for i in range(len(self.liceums)):

t = [0] \* len(self.districts)

t[self.find\_district(self.liceums[i])] = 1

y = [0] \* len(self.districts)

min\_dist = inf

min\_j = 0

for j in range(len(self.districts)):

Pj = self.get\_dist(self.liceums[i], self.districts[j], i, j)

if min\_dist > Pj:

min\_dist = Pj

min\_j = j

y[min\_j] = 1

if debug:

print(self.liceums[i][0])

print(f'\tCorrect: {self.liceums[i][3]}')

print(f'\tPredict: {self.districts[min\_j][0]}')

if y != t:

error += 1

self.correct(i, t, y)

return error

def get\_dist(self, liceum, district, i, j):

l\_coords = (liceum[1], liceum[2])

d\_coords = (district[1], district[2])

temp = self.W[j][i][0]\*l\_coords[0] + self.W[j][i][1]\*l\_coords[1]

return 0.5\*self.vector\_length(d\_coords) - temp

def vector\_length(self, vec):

return vec[0]\*vec[0] + vec[1]\*vec[1]

def correct(self, i, t, y):

for j in range(len(t)):

if t[j] == y[j]:

continue

self.W[j][i][0] += (t[j]-y[j])\*self.Rate\*self.liceums[i][1]

self.W[j][i][1] += (t[j]-y[j])\*self.Rate\*self.liceums[i][2]

def find\_district(self, liceum):

district = None

for j in range(len(self.districts)):

if liceum[3] == self.districts[j][0]:

return j

def main():

nn = NN(file\_liceums='./liceums', file\_districts='./districts')

nn.train(debug=False)

if \_\_name\_\_ == '\_\_main\_\_':

main()

Вывод программы

K: 0

E: 84/89

K: 1

E: 0/89

Список данных (в формате pandas.DataFrame)

name X Y okrug

0 Государственное бюджетное образовательное учре... 37.608043 55.721823 Центральный административный округ

1 Государственное бюджетное образовательное учре... 37.681319 55.676895 Южный административный округ

2 Государственное бюджетное образовательное учре... 37.649058 55.720986 Южный административный округ

3 Негосударственная международная образовательна... 37.570733 55.781992 Северный административный округ

4 Государственное бюджетное образовательное учре... 37.782105 55.743298 Восточный административный округ

.. ... ... ... ...

84 Государственное бюджетное образовательное учре... 37.552820 55.702144 Юго-Западный административный округ

85 Государственное бюджетное образовательное учре... 37.555175 55.702928 Юго-Западный административный округ

86 Государственное бюджетное образовательное учре... 37.554346 55.702554 Юго-Западный административный округ

87 Государственное бюджетное образовательное учре... 37.553572 55.702290 Юго-Западный административный округ

88 Государственное бюджетное образовательное учре... 37.555832 55.704718 Юго-Западный административный округ

[89 rows x 4 columns]

name X Y

0 Центральный административный округ 37.617634 55.755787

1 Зеленоградский административный округ 37.181683 55.988857

2 Новомосковский административный округ 37.352547 55.609756

3 Троицкий административный округ 37.326454 55.494152

4 Восточный административный округ 37.791595 55.781979

5 Западный административный округ 37.438660 55.673714

6 Северный административный округ 37.530670 55.832916

7 Южный административный округ 37.648773 55.644276

8 Северо-Восточный административный округ 37.621307 55.851421

9 Северо-Западный административный округ 37.422180 55.845253

10 Юго-Восточный административный округ 37.769623 55.693844

11 Юго-Западный административный округ 37.547150 55.605507