|  |  |  |  |
| --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования  Белорусский Государственный Университет Информатики и Радиоэлектроники | | | |
| Факультет | Компьютерных сетей и систем | | |
| Кафедра | Информатики  Дисциплина: Конструирование те технологии электронных вычислительных средств | | |
|  |  | | |
| **Отчет ПО Лабораторной Работе**  по курсу «Системы аналитического программирования» | | | |
| Студент:  гр. 758641  Ярош Г.И. | |  | Проверил:  Герман О.В. |
| Минск, 2018 | | | |

# КЛАСТЕРИЗАЦИЯ. МЕТОД К-СРЕДНИХ

## Цель

Изучить возможности языка Python и библиотеки SciPy для решения задачи кластеризации с помощью алгоритма K-средних.

## Теоретическая часть

Для поиска кластеров в неразмеченных данных часто используется алгоритм K-средних. Целью метода К-средних является минимизация расстояний от элементов кластеров до его центров, что отражено в следующей формуле:

Суть алгоритма заключается в следующем:

1. Изначально кластеры инициализируются случайным образом.
2. Вычисляется центр каждого кластера как среднее значение всех координат элементов.
3. Затем для каждого элемента считается расстояние от центра каждого кластера. Если координата находится ближе к другому кластеру, чем к текущему, то она попадает в него.
4. После вычисления расстояний и перераспределения элементов по кластерам происходит пересчет центров кластеров, т. е. переход к шагу 2.
5. Алгоритм продолжается до тех пор, пока кластеры не являются устойчивыми, т.е. ни один элемент не был перераспределен из одного кластера в другой.

## Реализация

Для поиска кластеров можно использовать библиотеку sklearn и модуль cluster. В нем содержится класс KMeans, который реализует метод К-средних. На вход он получает массив координат элементов и требуемое количество кластеров. В результате мы получаем центры найденных кластеров. Сами элементы кластеров можно получить с помошью функции pairwise\_distances\_argmin из модуля sklearn.metrics.pairwise, которая для каждого элемента находит ближайший центр кластера.

Для генерации данных можно использовать функцию make\_blobs из модуля sklearn.datasets.samples\_generator. Она принимает на вход центры кластеров, количество кластеров и дисперсию распределения координат элементов около центров. Возвращает массив координат элементов.

Исходный код программы приведен ниже:

*# -\*- coding: utf-8 -\*-*import numpy as np  
import matplotlib.pyplot as plt  
  
from sklearn.cluster import KMeans  
from sklearn.metrics.pairwise import pairwise\_distances\_argmin  
from sklearn.datasets.samples\_generator import make\_blobs  
  
  
*# генерация данных*def generate\_data(k):  
 centers = [[2, 2], [-2, -2], [-2, 2]]  
 x, \_ = make\_blobs(n\_samples=k, centers=centers, cluster\_std=2)  
 return x, len(centers)  
  
  
*# функция поиска кластеров*def find\_clusters(x, n):  
 *# поиск кластеров* k\_means = KMeans(n\_clusters=n, n\_init=100)  
 k\_means.fit(x)  
  
 *# определение элементов, попавших в каждый кластер* k\_means\_centers = np.sort(k\_means.cluster\_centers\_, axis=0)  
 k\_means\_indexes = pairwise\_distances\_argmin(x, k\_means\_centers)  
 return k\_means\_centers, k\_means\_indexes  
  
  
def main():  
 *# генерация исходных данных и поиск кластеров* x, n = generate\_data(3000)  
 centers, indices = find\_clusters(x, n)  
  
 fig = plt.figure()  
 ax1 = fig.add\_subplot(2, 1, 1)  
 ax2 = fig.add\_subplot(2, 1, 2)  
  
 *# график исходных данных* ax1.plot(  
 x[:, 0], x[:, 1], 'w',  
 markerfacecolor='black', marker='.', label='Members'  
 )  
  
 *# график кластеров* colors = ['green', 'blue', 'red']  
 markers = ['.', 'X', '^']  
 for k in range(n):  
 members = indices == k  
 ax2.plot(  
 x[members, 0], x[members, 1], 'w',  
 markerfacecolor=colors[k],   
 marker=markers[k],   
 label='{}th cluster'.format(k + 1)  
 )  
  
 ax2.legend(loc='best')  
 plt.show()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

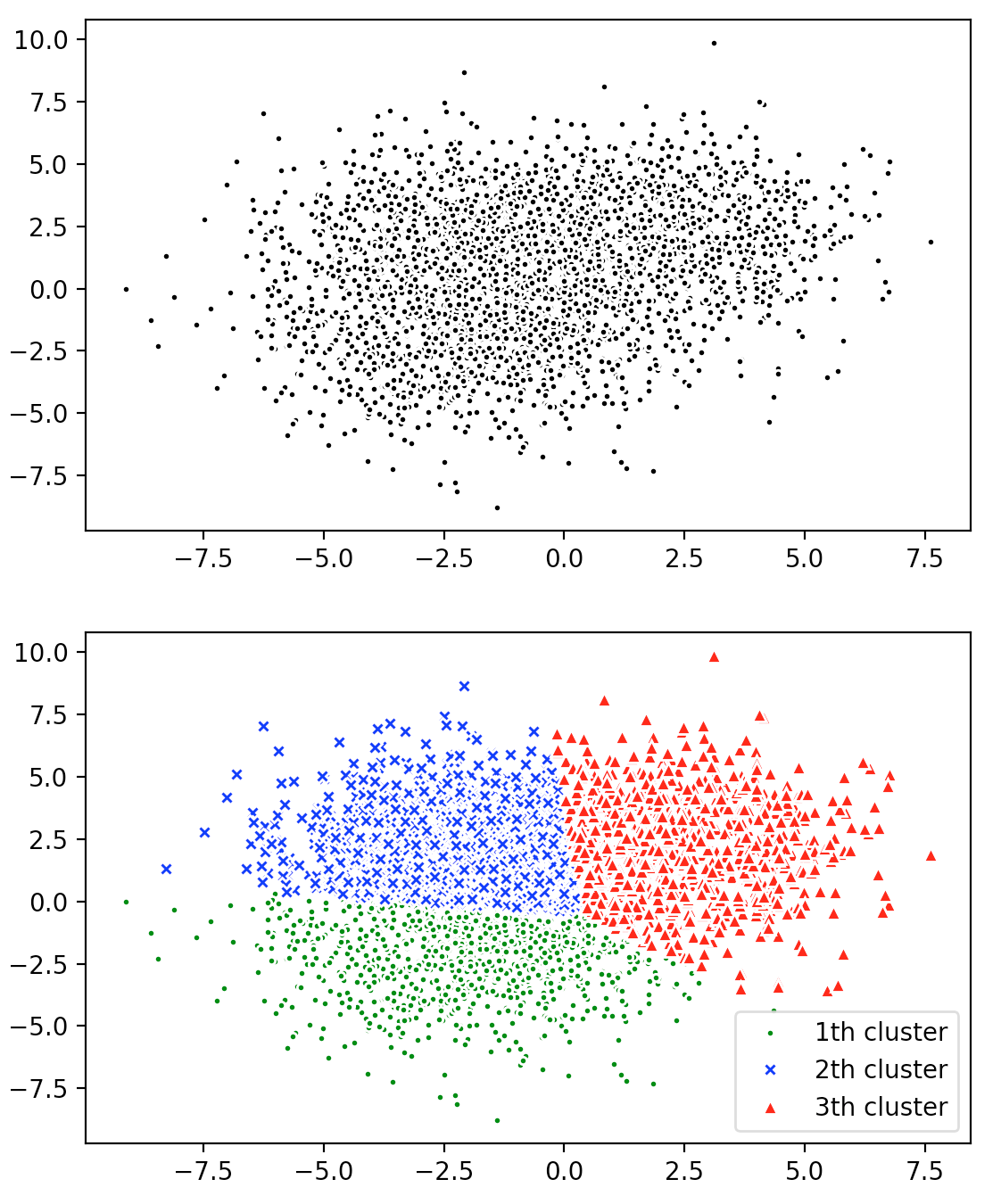
Полученные графики приведены на рисунке 1.   


Рис. 1. График координат исходных элементов и найденных кластеров.