Лабораторная работа №5: «Кластеризация»

Набор данных **ex6data1.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные X_1 и X_2 - координаты точек, которые необходимо кластеризовать.

Набор данных **bird_small.mat** представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит массив размером (16384, 3) - изображение 128x128 в формате RGB.

Задание.

1. Загрузите данные **ex6data1.mat** из файла.
2. Реализуйте функцию случайной инициализации K центров кластеров.
3. Реализуйте функцию определения принадлежности к кластерам.
4. Реализуйте функцию пересчета центров кластеров.
5. Реализуйте алгоритм K-средних.
6. Постройте график, на котором данные разделены на K=3 кластеров (при помощи различных маркеров или цветов), а также траекторию движения центров кластеров в процессе работы алгоритма
7. Загрузите данные **bird_small.mat** из файла.
8. С помощью алгоритма K-средних используйте 16 цветов для кодирования пикселей.
9. Насколько уменьшился размер изображения? Как это сказалось на качестве?
10. Реализуйте алгоритм K-средних на другом изображении.
11. Реализуйте алгоритм иерархической кластеризации на том же изображении. Сравните полученные результаты.
12. Ответы на вопросы представьте в виде отчета.

Реализация:

In[1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pnd
```

1.1 Загрузите набор данных ex6data1.mat из файла.

In[2]:

```
from scipy.io import loadmat

mat = loadmat('data/ex6data1.mat')
x = mat['X']
```

1.2 Реализуйте функцию случайной инициализации K центров кластеров

In[3]:

```
def init_centroids(X, K):
    m,n = X.shape[0], X.shape[1]
    centroids = np.zeros((K,n))

    for i in range(K):
        centroids[i] = X[np.random.randint(0 ,m+1),:]

    return centroids
```

1.3 Реализуйте функцию определения принадлежности к кластерам

In[4]:

```
def find_closes_clusters(X, centroids):
    K = centroids.shape[0]
    clusters = np.zeros(len(X), dtype=int)
    temp = np.zeros((centroids.shape[0],1))

    for i in range(X.shape[0]):
        for j in range(K):
            dist = X[i,:] - centroids[j,:]
            length = np.sum(dist**2)
            temp[j] = length

        clusters[i] = np.argmin(temp)

    return clusters
```

1.4 Реализуйте функцию пересчета центров кластеров

In[5]:

```
def update_centroids(X, clusters, K):
    new_centroids = np.zeros((K, X.shape[1]))
    for k in range(K):
        if len(X[clusters == k]) == 0:
            continue
        new_centroids[k] = X[clusters == k].mean(axis=0)

    return new_centroids
```

1.5 Реализуйте алгоритм K-средних

In[7]:

```
def k_means(X, centroids, clusters, K, num_iters):
    logs = [centroids]
    for i in range(num_iters):
        centroids = update_centroids(X, clusters, K)
        logs.append(centroids)
        clusters = find_closes_clusters(X, centroids)

    return clusters, logs
```

1.6. Постройте график, на котором данные разделены на K=3 кластеров (при помощи различных маркеров или цветов), а также траекторию движения центров кластеров в процессе работы алгоритма

In[8]:

```
def plot_k_means(X, clusters, cenroids_moving_logs):
    """
    plots the data points with colors assigned to each centroid
    """
    m, n = X.shape[0], X.shape[1]
    plt.scatter(X[:, 0], X[:, 1], c=clusters)

    clusters_centroids = [[], [], [], [], [], []]
    for centroids in cenroids_moving_logs:
        for idx, cl_centroid in enumerate(centroids):
            clusters_centroids[idx * 2].append(cl_centroid[0])
            clusters_centroids[idx * 2 + 1].append(cl_centroid[1])

    plt.plot(*clusters_centroids, marker='x')
    plt.show()

K = 3
initial_centroids = init_centroids(x, K)
clusters = find_closes_clusters(x, initial_centroids)
result_clusters, cenroids_moving_logs = k_means(x, initial_centroids, clus-
ters, 3, 50)
cenroids_moving_logs
plot_k_means(x, result_clusters, cenroids_moving_logs)
```