

Лабораторная работа №6 “Кластеризация”

Долматович Алина, 858641

Набор данных ex6data1.mat представляет собой файл формата *.mat (т.е. сохраненного из Matlab). Набор содержит две переменные X1 и X2 - 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.

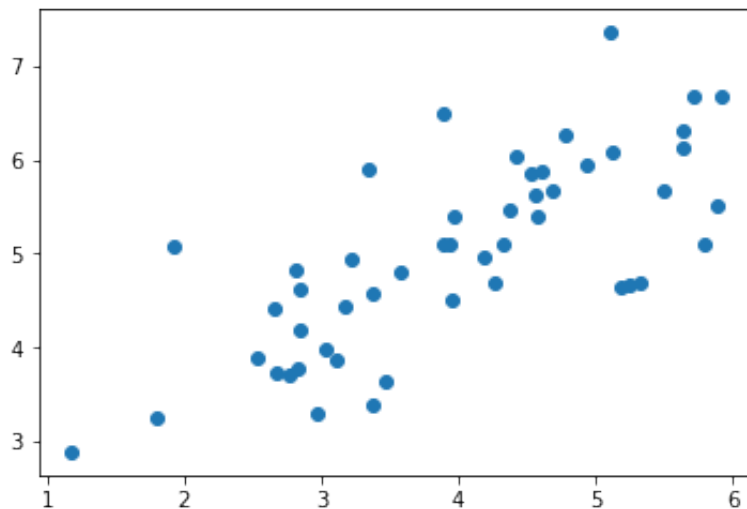
```
In [1]: 1 from scipy.io import loadmat
        2 import matplotlib.pyplot as pyplot
        3 from scipy.spatial.distance import cdist
        4 import numpy as np
        5 from scipy import misc
        6 from scipy.cluster import hierarchy
        7 from scipy.spatial.distance import pdist
        8 from scipy.cluster.hierarchy import fcluster
```

Загрузите данные ex6data1.mat из файла.

```
In [2]: 1 data = loadmat('ex6data1.mat')
        2
        3 x = data['X']
        4 print(x.shape)
```

(50, 2)

```
1 pyplot.scatter(x[:, 0], x[:, 1])
2 pyplot.show()
```



Реализуйте функцию случайной инициализации K центров кластеров.

```
1 def initialize(k=3, n=2, low=3, high=4):
2     return np.random.uniform(low=low, high=high, size=(k,n))
3
4 centroids = initialize()
5 print(centroids)
```

```
[ [ 3.46934373  3.50080345]
  [ 3.34081346  3.80715562]
  [ 3.27849946  3.15128936] ]
```

Реализуйте функцию определения принадлежности к кластерам.

```
1 def clusterMemberships(x, centroids):
2     distances = cdist(x, centroids)
3     memberships = distances.argmin(axis=1)
4     return memberships
5
6 memberships = clusterMemberships(x, centroids)
7 print(memberships)
```

[illegible]

Реализуйте функцию пересчета центров кластеров.

```
In [6]: 1 def recalculateCenters(x, centroids, memberships, k=3):
2         centroids = centroids.copy()
3         for index in range(k):
4             centroids[index, :] = np.mean(x[memberships == index, :],
5
6             return centroids
7
8 centroids = recalculateCenters(x, centroids, memberships)
9 print(centroids)
```

```
[[ 3.428684    3.51098061]
 [ 4.14810562  5.19315195]
 [ 1.98037302  3.14216042]]
```

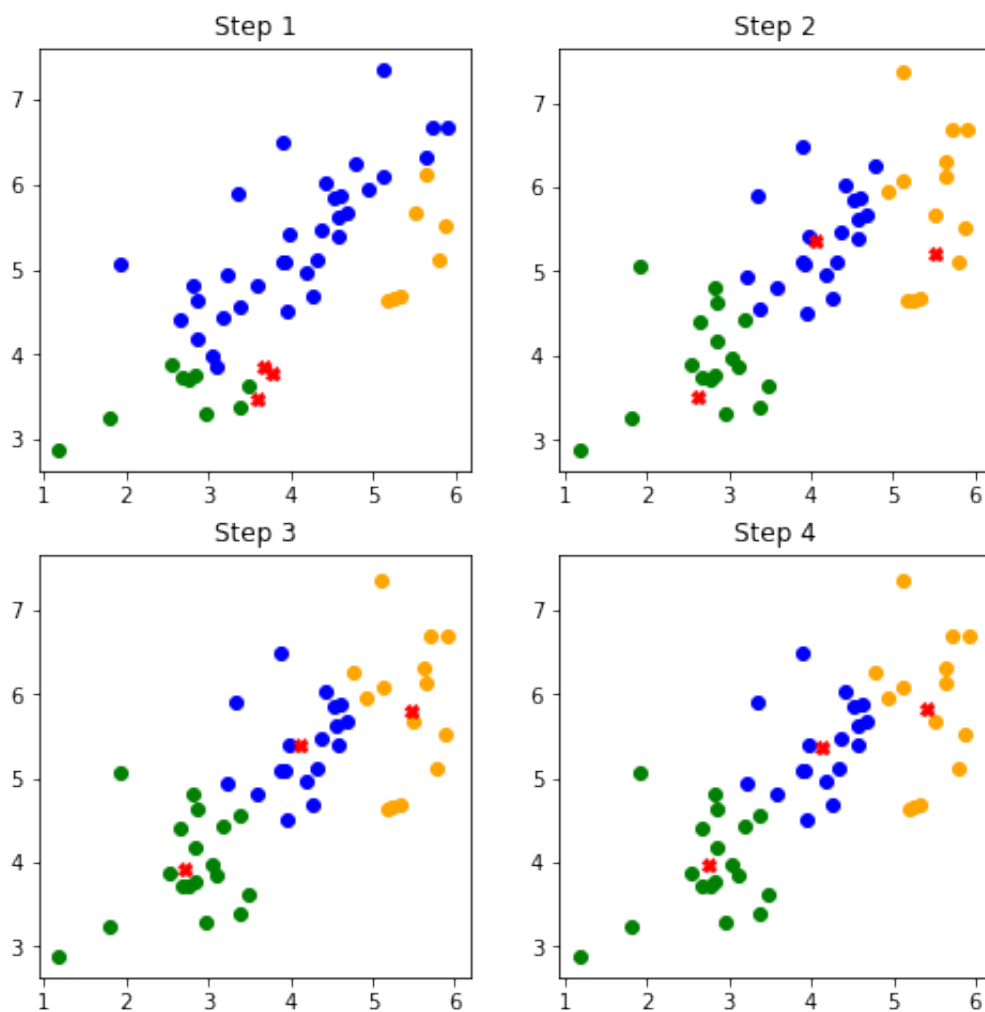
Реализуйте алгоритм К-средних.

```
In [7]: 1 def kMeans(x, centroids, k, n=2):
2         centroidsHistory = []
3         centroidsHistory.append(centroids)
4
5         iterations = 3
6
7         for i in range(iterations):
8             memberships = clusterMemberships(x, centroids)
9             centroids = recalculateCenters(x, centroids, memberships,
10             centroidsHistory.append(centroids)
11
12         return centroidsHistory
13
14 centroids = initialize(3, 2)
15 centroidsHistory = kMeans(x, centroids, 3)
16 print(centroidsHistory)
```

```
[array([[ 3.57640242,  3.48462973],
        [ 3.67674903,  3.87124636],
        [ 3.77520516,  3.79314668]]), array([[ 2.62353611,  3.5052595
1],
        [ 4.03767443,  5.35807228],
        [ 5.51007175,  5.20263767]]), array([[ 2.70689034,  3.9367106
],
        [ 4.12320662,  5.38602638],
        [ 5.46015406,  5.80736017]]), array([[ 2.74403119,  3.9716365
3],
        [ 4.1284461 ,  5.38337803],
        [ 5.41133404,  5.83928776]])]
```

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]: 1 def plotCentroidsHistory(centroidsHistory, k=3):
2         pyplot.figure(figsize=(8, 8))
3         for i in range(len(centroidsHistory)):
4             memberships = clusterMemberships(x, centroidsHistory[i])
5             pyplot.subplot(2, 2, i + 1)
6             colors = ["green", "blue", "orange"]
7             for kIndex, color in zip(range(k), colors):
8                 pyplot.scatter(x[memberships == kIndex, 0], x[memberships == kIndex, 1], color)
9             pyplot.plot(centroidsHistory[i][:, 0], centroidsHistory[i][:, 1], color='red', marker='x')
10            pyplot.title('Step {}'.format(i + 1));
11
12         pyplot.show()
13
14 plotCentroidsHistory(centroidsHistory)
```



Загрузите данные bird_small.mat из файла.

```
In [9]: 1 birdData = loadmat('bird_small.mat')
        2
        3 aOriginal = birdData['A']
        4 print(aOriginal.shape)
```

```
(128, 128, 3)
```

С помощью алгоритма К-средних используйте 16 цветов для кодирования пикселей.

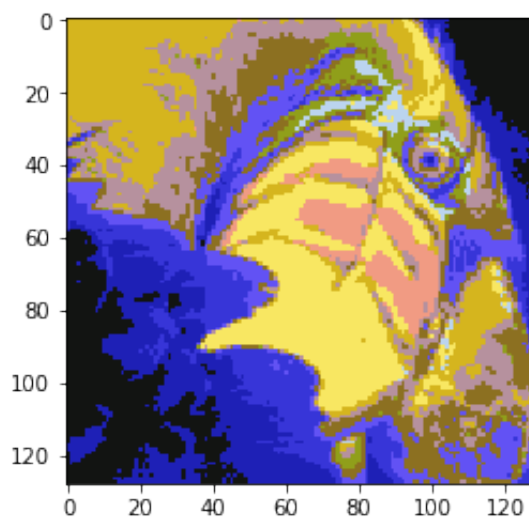
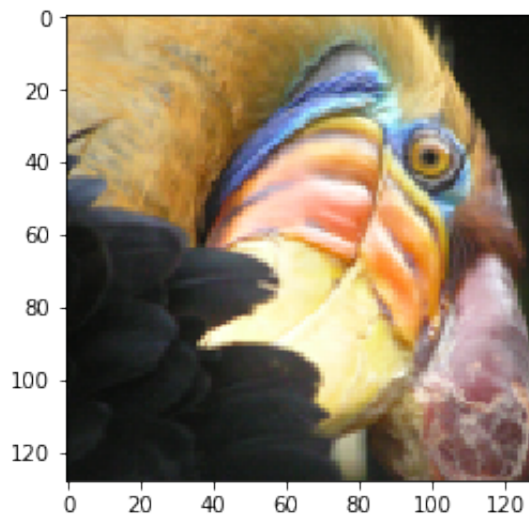
```
In [10]: 1 def distance(x1, y1, x2, y2):
        2     dist = np.square(x1 - x2) + np.square(y1 - y2)
        3     dist = np.sqrt(dist)
        4     return dist
        5
        6 def kMeansFor(image, clusters):
        7     iterations = 10
        8     points = np.reshape(image, (image.shape[0] * image.shape[1], image.shape[2]))
        9     m, n = points.shape
       10     index = np.zeros(m)
       11
       12     means = initialize(k=clusters, n=image.shape[2], low=0, high=255)
       13
       14     while(iterations > 0):
       15         for j in range(len(points)):
       16             minv = 1000
       17             temp = None
       18
       19             for k in range(clusters):
       20                 x1 = points[j, 0]
       21                 y1 = points[j, 1]
       22                 x2 = means[k, 0]
       23                 y2 = means[k, 1]
       24
       25                 if(distance(x1, y1, x2, y2) < minv):
       26                     minv = distance(x1, y1, x2, y2)
       27                     temp = k
       28                     index[j] = k
       29
       30             for k in range(clusters):
       31                 sumx = 0
       32                 sumy = 0
       33                 count = 0
       34
       35             for j in range(len(points)):
       36                 if(index[j] == k):
       37                     sumx += points[j, 0]
       38                     sumy += points[j, 1]
       39                     count += 1
       40
```

```
41         if(count == 0):
42             count = 1
43
44             means[k, 0] = float(sumx / count)
45             means[k, 1] = float(sumy / count)
46
47             iterations -= 1
48
49     return means, index
```

Насколько уменьшился размер изображения? Как это сказалось на качестве?

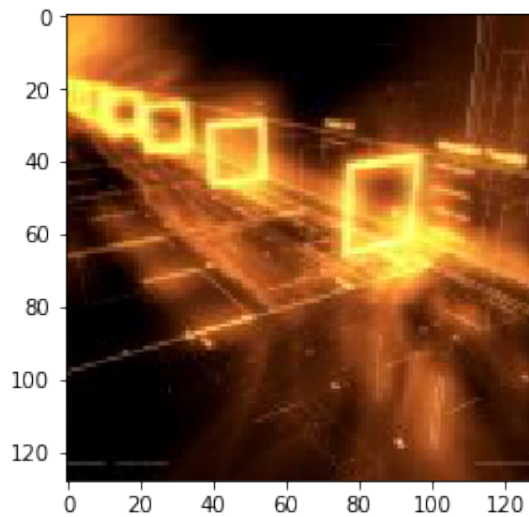
```
In [11]: 1 def compress_image(means, index, size):
2         centroid = np.array(means)
3         recovered = centroid[index.astype(int), :]
4         recovered = np.reshape(recovered, (128, 128, 3))
5         pyplot.imshow(recovered)
6         pyplot.show()
7
8     k = 16
9     image = aOriginal / 255.
10    means, index = kMeansFor(image, k)
```

```
In [12]: 1 pyplot.imshow(aOriginal)
          2 pyplot.show()
          3
          4 compress_image(means, index, image)
```

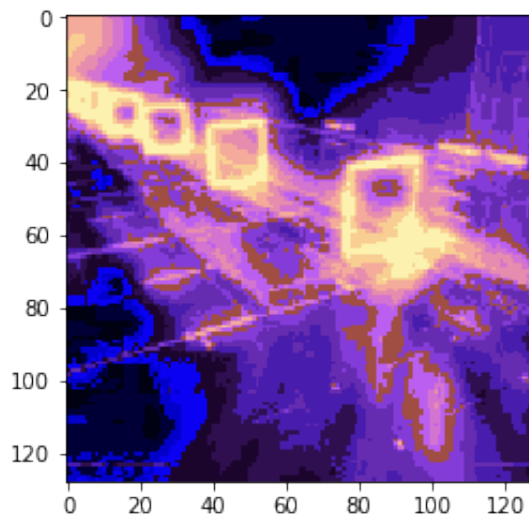


Реализуйте алгоритм К-средних на другом изображении.

```
In [13]: 1 testImage = misc.imread('testImage.jpg')
2         pyplot.imshow(testImage)
3         pyplot.show()
4
5         testImage = testImage / 255.
6         print(testImage.shape)
7
8         means, index = kMeansFor(testImage, k)
9         compress_image(means, index, testImage)
```



(128, 128, 3)

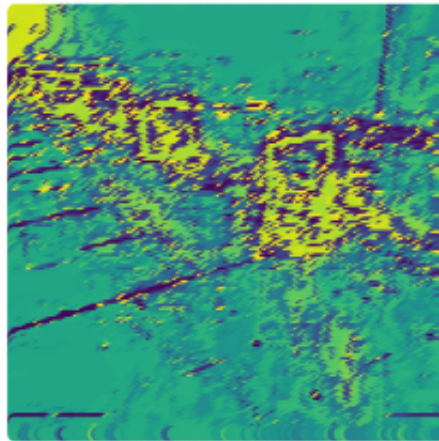


Реализуйте алгоритм иерархической кластеризации на том же изображении. Сравните полученные результаты.


```

In [14]: nts1 = np.reshape(testImage, (testImage.shape[0] * testImage.shape[1], t
2
3         distance_mat = pdist(points)
4
5         hierarchy.linkage(distance_mat, 'single')
6         _d = .1
7         while max_d > 0.005:
8             max_d *= .5
9             clusters = fcluster(Z, max_d, criterion='distance')
10            meshx, meshy = np.meshgrid(np.arange(128), np.arange(128))
11
12            plt.axis('equal')
13            plt.axis('off')
14            plt.scatter(meshx, -(meshy - 128), c=clusters.reshape(128, 128))
15            plt.show()

```



Вывод

Кластеризация — задача группировки множества объектов на подмножества (кластеры) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию.

Задача кластеризации относится к классу задач обучения без учителя.

Часто в алгоритмах машинного обучения возникает необходимость классифицировать данные. Каждый объект данных представляется как вектор (точка) в p -мерном пространстве (упорядоченный набор p чисел). Каждая из этих точек принадлежит только одному из двух классов. Вопрос состоит в том, можно ли разделить точки гиперплоскостью размерности $(p-1)$. Это — типичный случай линейной разделимости. Искомых гиперплоскостей может быть много, поэтому полагают, что максимизация зазора между классами способствует более уверенной классификации. То есть, можно ли найти такую гиперплоскость, чтобы расстояние от неё до ближайшей точки было максимальным. Это эквивалентно[1] тому, что сумма расстояний до гиперплоскости от двух ближайших к ней точек, лежащих по разные стороны от неё, максимально. Если такая гиперплоскость существует, она называется оптимальной разделяющей гиперплоскостью, а соответствующий ей линейный классификатор называется оптимально разделяющим классификатором.