Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра информатики

ЛабоРАТОРНАЯ РАБОТА №6

«Кластеризация»

Выполнил: Яловчук Валерий Валерьевич

магистрант кафедры информатики

группа №858641

Проверил: доцент, кандидат технических наук Стержанов Максим Валерьевич

Минск 2019

ХОД РАБОТЫ

**Данные.**

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.

**Выполнение:**

1. Загрузите данные ex6data1.mat из файла:

file\_path = 'ex6data1.mat'

dataset = sio.loadmat(file\_path)

X = dataset["X"]

1. Реализуйте функцию случайной инициализации K центров кластеров:

def generate\_k\_rand\_centroids(k, n, min=0, max=8):

centroids = []

for i in range(k):

centroid = []

for j in range(n):

centroid.append(random.randint(min, max))

centroids.append(centroid)

return np.array(centroids)

K = 3

initial\_centroids = generate\_k\_rand\_centroids(K, 2)

print(initial\_centroids)

initial\_centroids = np.array([[3, 3], [6, 2], [8, 5]]) # use mock

3. Реализуйте функцию определения принадлежности к кластерам:

print(find\_closest\_centroid(np.array([[3.38156267, 3.38911268]]), initial\_centroids))

centroids = find\_closest\_centroid(X, initial\_centroids)

Результат выполнения:

[[1.]]

4. Реализуйте функцию пересчета центров кластеров:

def find\_closest\_centroid(X, centroids):

K = centroids.shape[0]

idx = np.zeros((X.shape[0], 1))

temp = np.zeros((centroids.shape[0], 1))

for i in range(X.shape[0]):

for j in range(K):

dist = X[i] - centroids[j]

temp[j] = np.sum(dist\*\*2) # a^2 + b^2

idx[i] = np.argmin(temp) + 1

return idx

compute\_centroids(X, centroids, K)

5. Реализуйте алгоритм K-средних:

def k\_means(X, idx, K, num\_iters):

for i in range(num\_iters):

# Compute the centroids mean

centroids = compute\_centroids(X, idx, K)

# assign each training example to the nearest centroid

idx = find\_closest\_centroid(X, centroids)

return [centroids, idx]

k\_means(X, centroids, K, 10)

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:

initial\_centroids = generate\_k\_rand\_centroids\_from\_dataset(X, K)

plot\_k\_means(X, initial\_centroids, K, 10)

Результат выполнения:

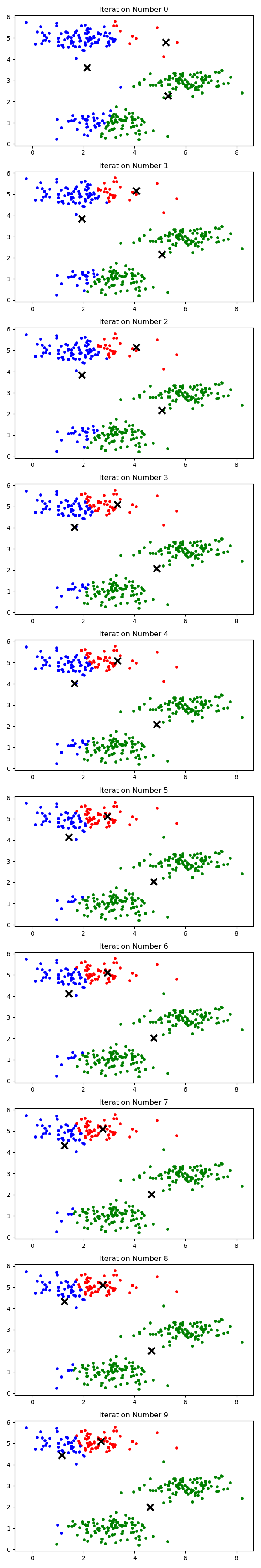
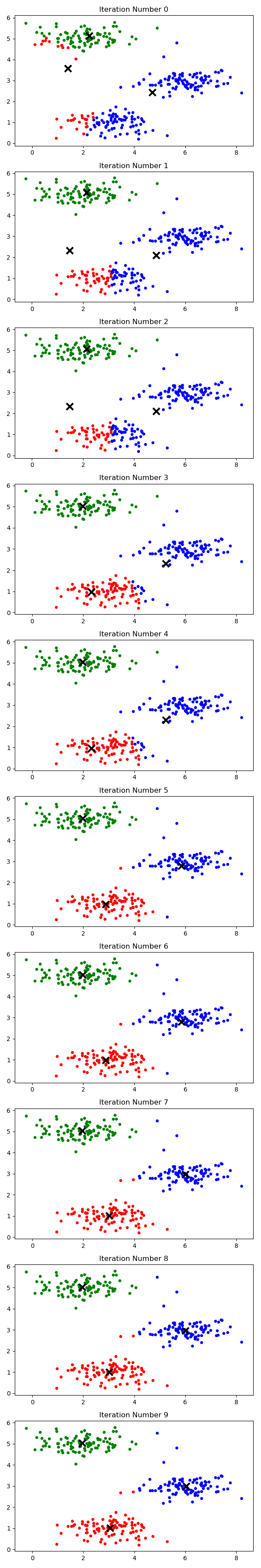
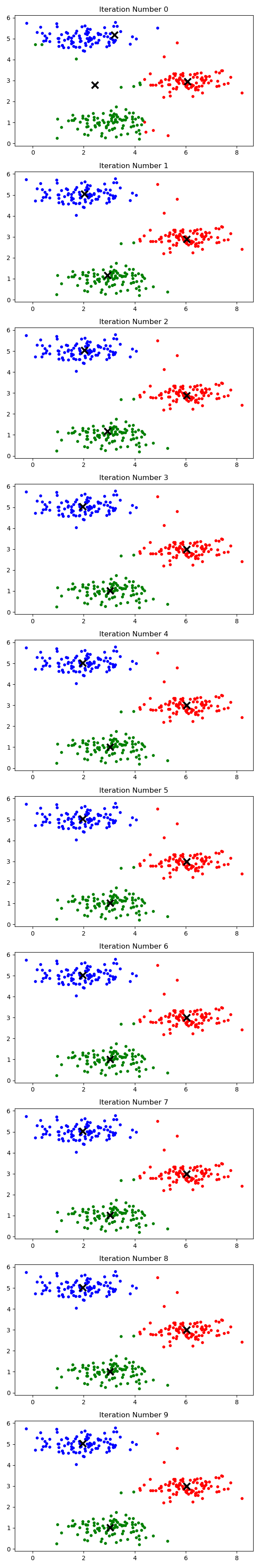


Рисунок 1 – трактория движения центров кластеров

K-means алгоритм чувствителен к выбору начальных координат кластеров. Алгоритм сходится в локальном минимуме, который может совпасть с глобальным, что и произошло в первых двух запусках. При третьем запуске выбор начальных координат оказался ненудачным.

7. Загрузите данные bird\_small.mat из файла:

file\_path = 'bird\_small.mat'

dataset = sio.loadmat(file\_path)

A = dataset["A"]

# preprocess and reshape the image

X = (A/255).reshape(128\*128, 3)

8. С помощью алгоритма K-средних используйте 16 цветов для кодирования пикселей:

K = 16

num\_iters = 10

initial\_centroids = generate\_k\_rand\_centroids\_from\_dataset(X, K, 0, 16384)

idx = find\_closest\_centroid(X, initial\_centroids)

[centroids, idx] = k\_means(X, idx, K, num\_iters)

9. Насколько уменьшился размер изображения? Как это сказалось на качестве?

X\_recovered = X.copy()

for i in range(1, K+1):

X\_recovered[(idx == i).ravel(), :] = centroids[i-1]

# Reshape the recovered image into proper dimensions

X\_recovered = X\_recovered.reshape(128, 128, 3)

# Display the image

fig, ax = plt.subplots(1, 2)

ax[0].imshow(X.reshape(128, 128, 3))

ax[1].imshow(X\_recovered)

plt.show()

Результат выполнения:

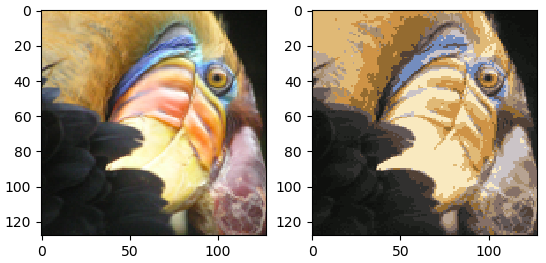


Рисунок 2 – изображение до (слева) и после (справа) кодирования цвета пикселей с помощью k-means с 16 кластерами

Размер изображения уменьшился в ~10 раз. После жатия изображение потеряло детализацию. Это ожидаемо, количество возможных цветов уменьшилось с 256 ^ 3 до 16.

10. Реализуйте алгоритм K-средних на другом изображении:

file\_path = 'bird.png'

matrix = imageio.imread(file\_path)

X = (matrix/255).reshape(443\*590, 3)

K = 16

num\_iters = 10

initial\_centroids = generate\_k\_rand\_centroids\_from\_dataset(X, K, 0, 16384)

idx = find\_closest\_centroid(X, initial\_centroids)

[centroids, idx] = k\_means(X, idx, K, num\_iters)

X\_recovered = X.copy()

for i in range(1, K+1):

X\_recovered[(idx == i).ravel(), :] = centroids[i-1]

# Reshape the recovered image into proper dimensions

X\_recovered = X\_recovered.reshape(443, 590, 3)

# Display the image

fig, ax = plt.subplots(1, 2)

ax[0].imshow(X.reshape(443, 590, 3))

ax[1].imshow(X\_recovered)

plt.show()

Результат выполнения:

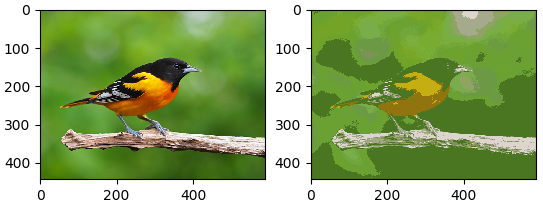


Рисунок 3 – собственное изображение до (слева) и после (справа) кодирования цвета пикселей с помощью к-средних с 16 кластерами с 1 итерацией

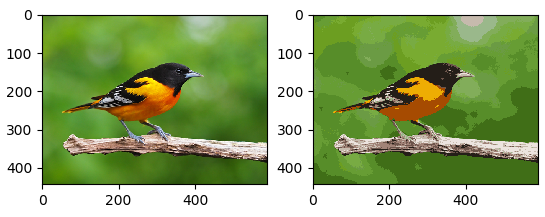


Рисунок 4 – собственное изображение до (слева) и после (справа) кодирования цвета пикселей с помощью к-средних с 16 кластерами с 10 итерациями

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

file\_path = 'bird-small.png'

img = imageio.imread(file\_path)

plt.imshow(img)

plt.show()

img = img / 255 # feature scaling

points = np.reshape(img, (img.shape[0] \* img.shape[1], img.shape[2]))

distance\_mat = pdist(points)

Z = hierarchy.linkage(distance\_mat, 'single')

max\_d = .3

while max\_d > 0.005:

max\_d \*= .5

print(max\_d)

clusters = fcluster(Z, max\_d, criterion='distance')

meshx, meshy = np.meshgrid(np.arange(128), np.arange(128))

plt.axis('equal')

plt.axis('off')

plt.scatter(meshx, -(meshy - 128), c=clusters.reshape(128, 128), cmap='inferno', marker=',')

plt.show()

Результат выполнения:

0.15

0.075

0.0375

0.01875

0.009375

0.0046875

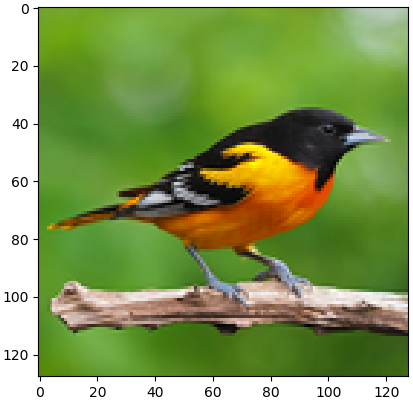
****

Рисунок 5 – исходное изображение

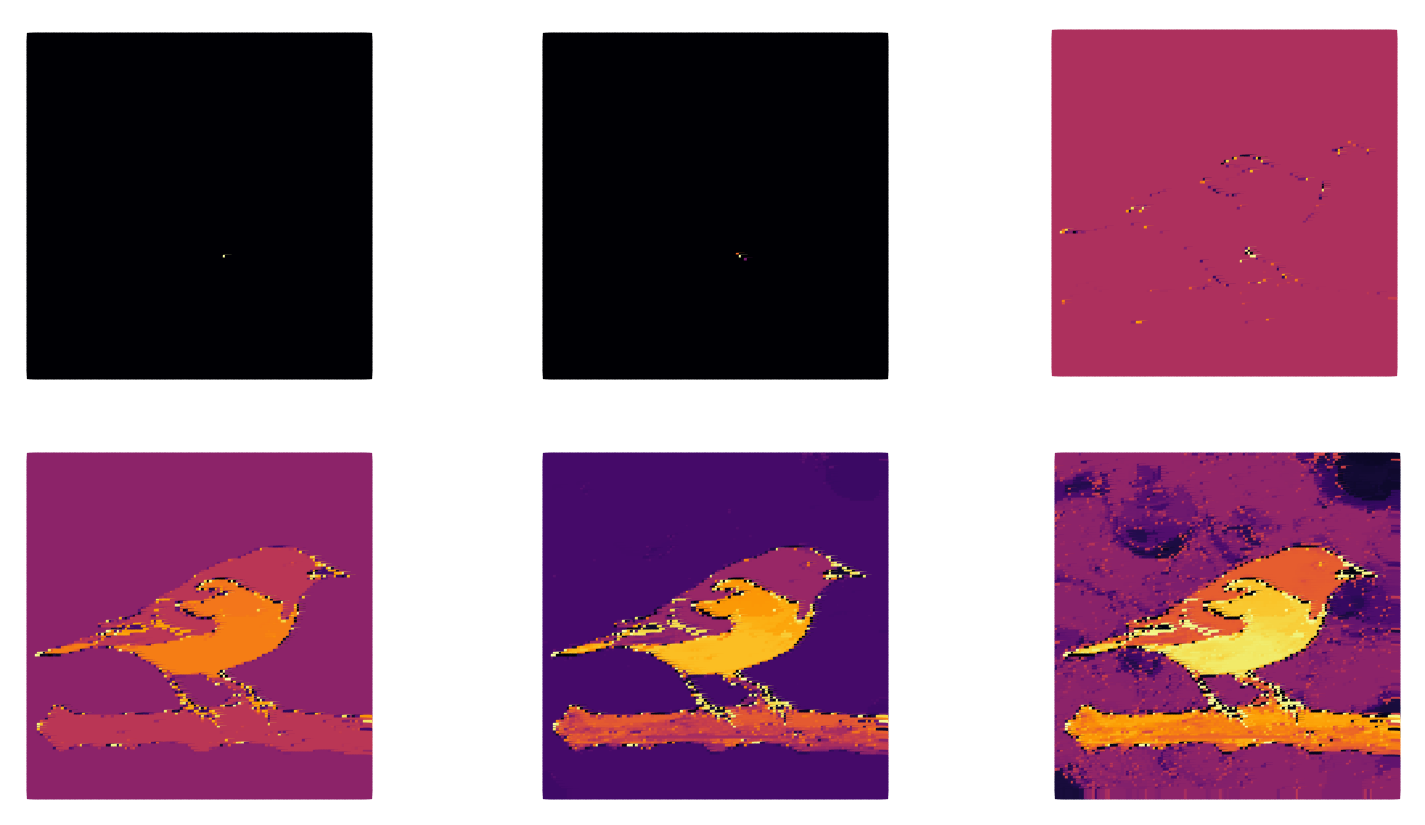
****

Рисунок 6 – результат работы алгоритма иерархической кластеризации

K-means определяет кластеры итеративно сходясь в локальном минимуме пресделую optimization objective. Этот алгоритм, даже при удачных начальных кластерах не всегда работает.

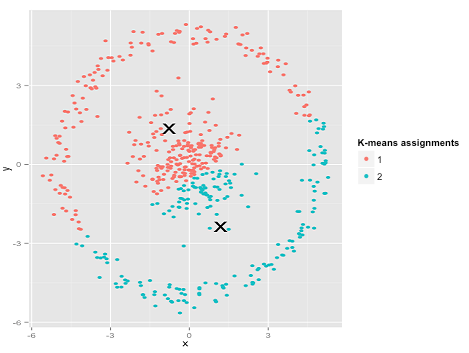


Рисунок 7 – ошибочная кластеризация K-means алгоритмом

В отличии от K-means, hierarchical clustering алгоритм в данном примере выше не допустит ошибки. Hierarchical clustering разделяют на кластеры по top-down and bottom-up подходам. В bottom-up подходе происходит соединение 2 близжайших элементов.

В текущем примере оба алгоритма отработали схоже, успешно сжав изображение.