HW1 Q2 K-medoids Arjun Singh

May 24, 2020

[113]:
```
import numpy as np
import time
```

[114]:
```
def init_medoids(X, k):
#   Randomly select k points from the image as the starting centers
    centers = np.random.choice(len(X),k,replace=False)
#     print(np.repeat(centers,k))
#     temp = np.repeat(centers,k)
#     return X[temp,:]
    return X[centers,:]
```

[115]:
```
# Compute  distance. I've included the metric p in the input to allow various␣
 ↪norms to compute the
# distance (rather than just Euclidean)
def compute_distance(X, centers, p):
    m = len(X)

    center_shape = centers.shape
    if len(center_shape)==1:
        centers = centers.reshape((1,len(centers)))

    s=len(centers)

    S = np.empty((m,s))
    for i in range(m):
        S[i,:] = np.linalg.norm(X[i,:]-centers,ord=p,axis=1)**p
    return S
```

[116]:
```
# Another approach to calculating distance between points in cluster. Avoids␣
 ↪the loop in the previous distance
# calculation approach.
def distance_calc (cluster_points,center,p,k):
    dist = []
    label = []
    min_dist = []

    main_array = list(range(k))
```

```
    pixel_temp = np.tile(center,(len(cluster_points),1))

    main_array= np.linalg.norm(cluster_points - pixel_temp, ord=p,axis=1)
    min_dist = main_array

    return min_dist
```

[117]:
```
def assign_cluster_labels(S):
    return np.argmin(S,axis=1)
```

[118]:
```
# Updates the cluster centers by finding the point with the least distance to␣
 ↪all other points.
def update_centers(X, centers, p,k):

    S = compute_distance(X, centers,p)
    labels = assign_cluster_labels(S)

    curr_centers = centers
#   Iterate over all clusters
    for i in set(labels):
        cluster_points = X[labels==i]
        cluster_points = np.unique(cluster_points,axis=0)
        avg_distance = np.sum(distance_calc(cluster_points, centers[i],p,k))
#       Iterate over all points in each cluster
        for points in cluster_points:
            new_distance = np.sum(distance_calc(cluster_points,points,p,k))
#           If distance from current point to all other points in the cluster␣
 ↪is lower than previous minimum
#           then update cluster center to the new point

            if new_distance < avg_distance:
                avg_distance = new_distance
                curr_centers[i] = points

    return curr_centers
```

[119]:
```
# Calculate the within clusters sum of squares
def WCSS(S):
    return np.sum(np.amin(S,axis=1))
```

[121]:
```
def kmedoids(X, k,p,starting_centers=None,max_steps=np.inf):
    start = time.time()
#   Begin by initializing starting points
    if starting_centers is None:
        centers = init_medoids(X, k)
    else:
        centers = starting_centers
```

```
    converged = False
    labels = np.zeros(len(X))
    i = 1
    wc= 99999999999
#   Begin loop for algorithm
    while (not converged) and (i <= max_steps):
        oldwcss = wc
        old_centers = centers

        S = compute_distance(X, old_centers,p)

        labels = assign_cluster_labels(S)

        centers = update_centers(X, centers,p,k)

        wc = WCSS(S)
#       If current step's WCSS is less than a 5% imporvement over previous␣
 ↪step, terminate
        if wc > 0.95*oldwcss:
            converged = True
        else:
            converged = False
        print ("iteration", i, "WCSS = ", WCSS (S))
        i += 1

    stop = time.time()
    print("Time taken = {:.2f} seconds".format(stop-start))
    return labels
```

[62]:
```
from matplotlib.pyplot import imshow
%matplotlib inline
def display_image(arr):
    """
    display the image
    input : 3 dimensional array
    """
    arr = arr.astype(dtype='uint8')
    img = Image.fromarray(arr, 'RGB')
    imshow(np.asarray(img))
```

[63]:
```
from PIL import Image
def read_img(path):
    """
    Read image and store it as an array, given the image path.
    Returns the 3 dimensional image array.
    """
```

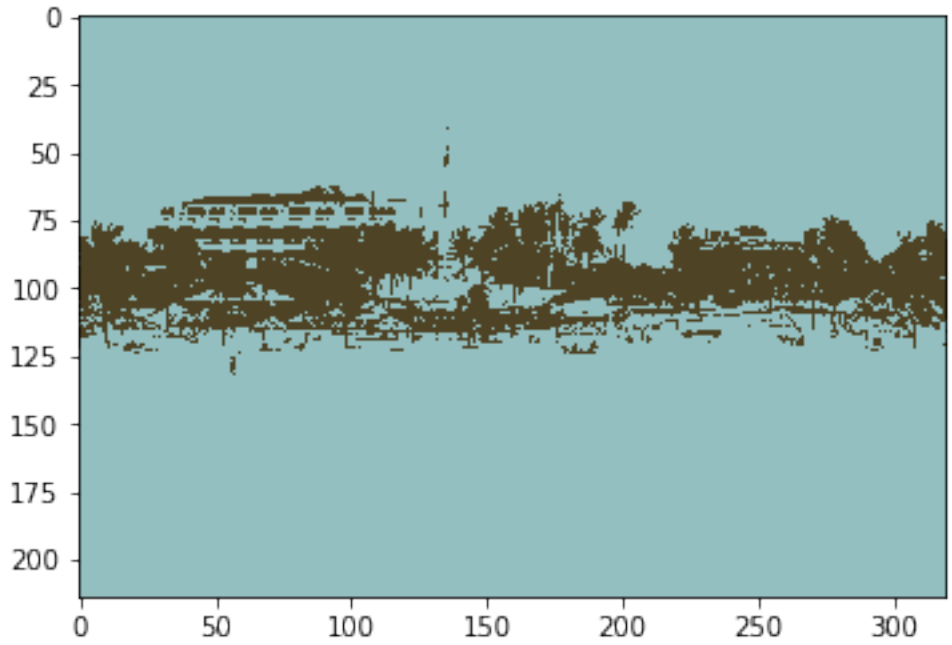
```
    img = Image.open(path)
    img_arr = np.array(img, dtype='int32')
#     img_arr = np.array(img.getdata())

    img.close()
    return img_arr
```

[64]:
```
def runKMedoidsAlgorithm(pixels,k):
    print("Running k-medoids algorithm...")
#   Read in image
    image = read_img(pixels)
    r, c, l = image.shape
#   Flatten image
    img_reshaped = np.reshape(image, (r*c, l), order="C")
#   Choose norm criteria
    p=2
#   If the value k is too high, reduce it
    if k>= (len(img_reshaped)/3):
        k = (len(img_reshaped)/3)

#   Run algorithm
    labels = kmedoids(img_reshaped, k,p, starting_centers=None)
    ind = np.column_stack((img_reshaped, labels))
    centers = {}
    for i in set(labels):
        c = ind[ind[:,3] == i].mean(axis=0)
        centers[i] = c[:3]

    img_clustered = np.array([centers[i] for i in labels])
    r, c, l = image.shape
    img_disp = np.reshape(img_clustered, (r, c, l), order="C")

    print('Image with k = ' + str(k) + " clusters...")
    display_image(img_disp)

    print("The labels are: {}".format(labels.T))
    print("The cluster centers are {}".format(centers))

    return labels,centers
```

[104]:
```
lab, cen = runKMedoidsAlgorithm("data/beach.bmp",2)
```

```
Running k-medoids algorithm…
iteration 1 WCSS =  1294971087.0
iteration 2 WCSS =  437519881.0
iteration 3 WCSS =  412498211.0
iteration 4 WCSS =  412002293.0
```

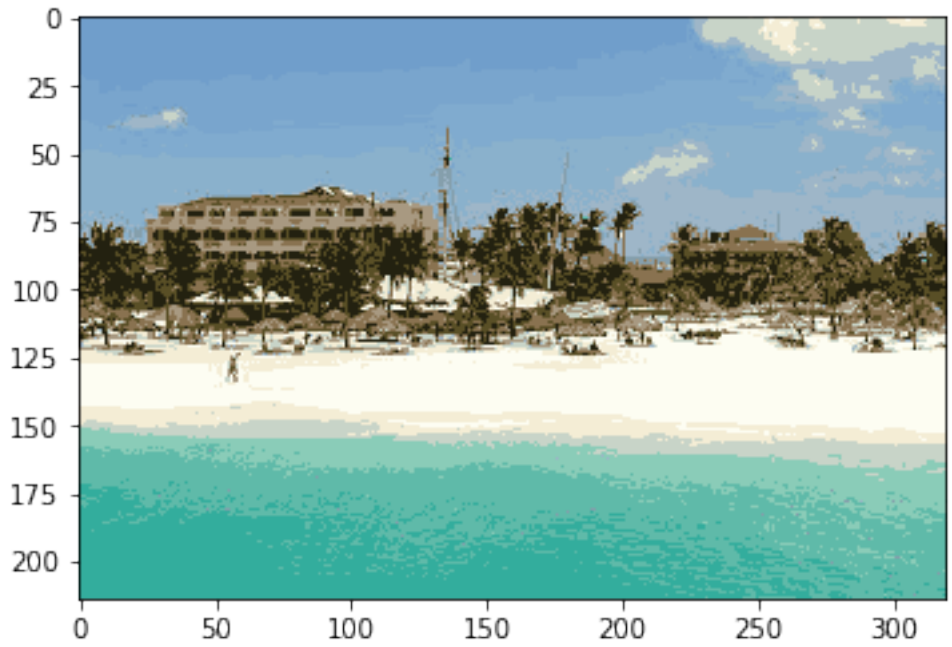
```
Time taken = 85.63 seconds
Image with k = 2 clusters…
The labels are: [0 0 0 … 0 0 0]
The cluster centers are {0: array([147.16211718, 191.53710217, 193.77184524]),
1: array([78.19784208, 67.69475233, 36.66287396])}
```



[105]:
```
lab, cen = runKMedoidsAlgorithm("data/beach.bmp",16)
```

```
Running k-medoids algorithm…
iteration 1 WCSS =  39547446.0
iteration 2 WCSS =  29251741.0
iteration 3 WCSS =  28094872.0
Time taken = 25.94 seconds
Image with k = 16 clusters…
The labels are: [ 9  9  9 … 14 14 14]
The cluster centers are {0: array([138.93415033, 177.11029412, 204.60669935]),
1: array([128.04694206, 171.7360515 , 207.2194206 ]), 2: array([158.61106899,
182.99620925, 200.34571645]), 3: array([133.44534413, 159.74898785,
172.53846154]), 4: array([138.28383562, 204.64356164, 184.32027397]), 5:
array([168.84196185, 153.09264305, 129.19663942]), 6: array([244.19209848,
237.29716576, 213.98854853]), 7: array([121.14083333, 165.68666667,
205.32805556]), 8: array([ 51.49277347, 173.9493382 , 157.42933212]), 9:
array([112.45182434, 158.05724274, 203.68756828]), 10: array([200.41001221,
212.13626374, 200.37338217]), 11: array([89.62761613, 79.46579888,
38.39841756]), 12: array([253.87523681, 253.52002706, 242.33667118]), 13:
array([35.378612  , 33.70560632, 16.00740924]), 14: array([ 95.11442588,
```

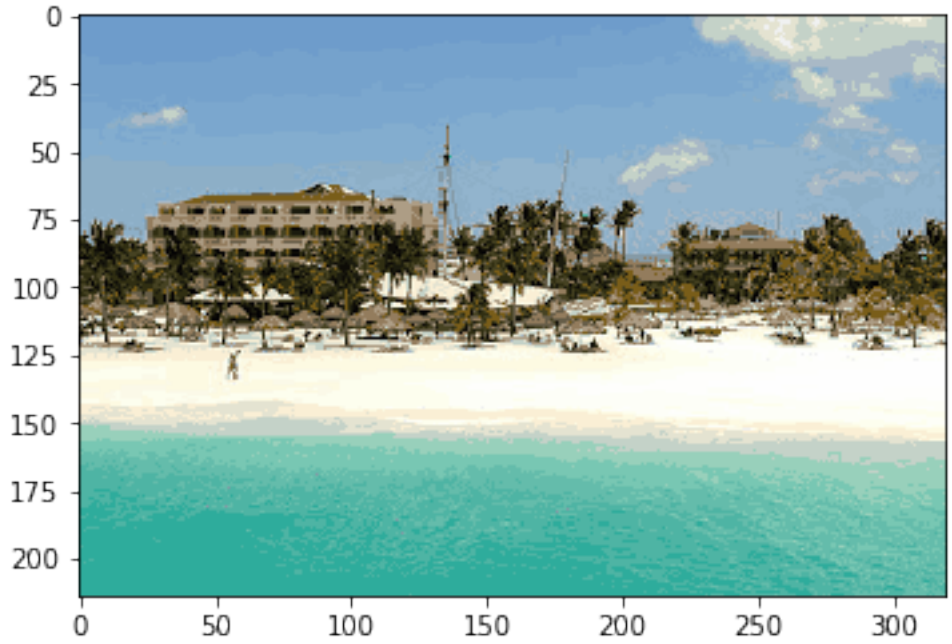
```
186.68443384, 169.57703527]), 15: array([137.79430789, 114.00258732,
80.36578266])}
```



[106]:
```
lab, cen = runKMedoidsAlgorithm("data/beach.bmp",32)
```

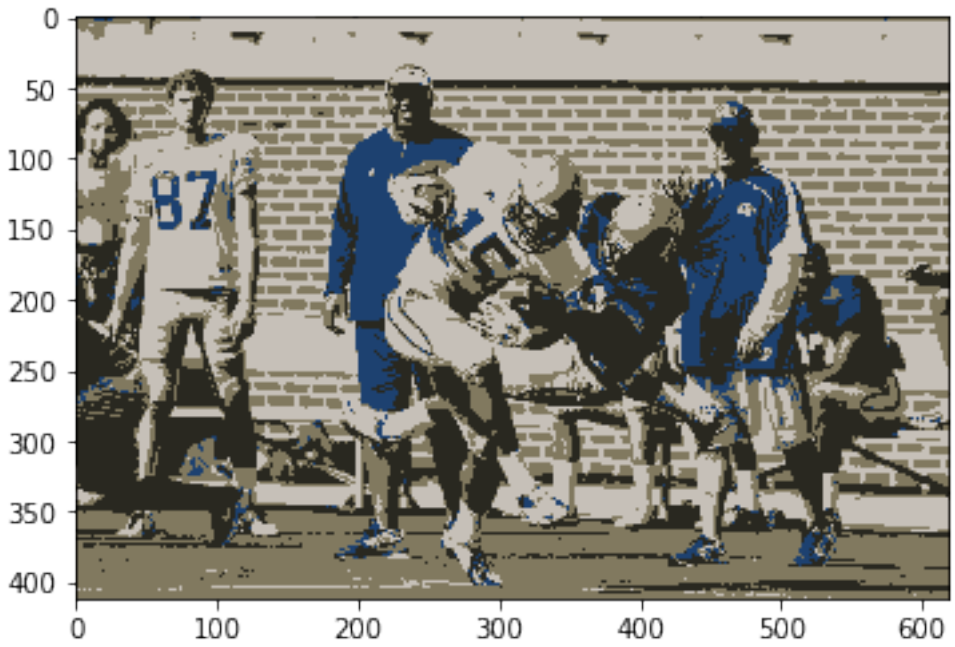
```
Running k-medoids algorithm…
iteration 1 WCSS =  40365040.0
iteration 2 WCSS =  25836544.0
iteration 3 WCSS =  21509027.0
iteration 4 WCSS =  18861152.0
iteration 5 WCSS =  17949460.0
Time taken = 37.14 seconds
Image with k = 32 clusters…
The labels are: [9 9 9 … 6 6 6]
The cluster centers are {0: array([102.61856254,  93.72778721,  72.14714205]),
1: array([123.60374314, 167.72668603, 206.10519522]), 2: array([133.08716707,
154.8716707 , 166.63680387]), 3: array([253.91415577, 254.14383172,
238.14837976]), 4: array([169.11556728, 185.73192612, 198.78416887]), 5:
array([253.593361  , 250.41138115, 226.5429757 ]), 6: array([100.45764167,
188.36119061, 171.1044648 ]), 7: array([141.13195876, 175.62852234, 199.2628866
]), 8: array([3.29399586, 2.33747412, 4.94202899]), 9: array([110.36245016,
156.20188474, 203.12794491]), 10: array([228.62790698, 227.20237506,
209.07174666]), 11: array([120.51625306, 199.07375044, 179.69136666]), 12:
array([254.45564893, 254.60037348, 252.49579832]), 13: array([196.77697161,
210.73911672, 200.34794953]), 14: array([146.29267613, 180.89334447,
205.13951546]), 15: array([70.88827586, 66.62482759, 34.08367816]), 16:
```

```
array([48.80955088, 46.22285389, 18.72370665]), 17: array([254.95708155,
254.79828326, 254.98998569]), 18: array([152.08618899, 208.5768432 ,
187.72689512]), 19: array([131.00260473, 173.88083351, 207.39830692]), 20:
array([240.41935484, 241.73387097, 241.28225806]), 21: array([253.9470437 ,
254.50796915, 247.85141388]), 22: array([121.25098296,  93.51572739,
25.62057667]), 23: array([147.44496991, 124.08340499,  93.03568358]), 24:
array([251.81137725, 249.81137725, 232.24700599]), 25: array([117.06461445,
162.18957635, 204.65650716]), 26: array([250.5407226 , 239.955297  ,
213.11757502]), 27: array([ 78.53684448, 180.65257685, 164.23915009]), 28:
array([27.68958743, 27.79174853, 14.38212181]), 29: array([14.37115839,
12.356974  ,  7.64539007]), 30: array([173.98724348, 157.92956184, 133.5890183
]), 31: array([ 46.63374007, 172.5704048 , 156.07883014])}
```



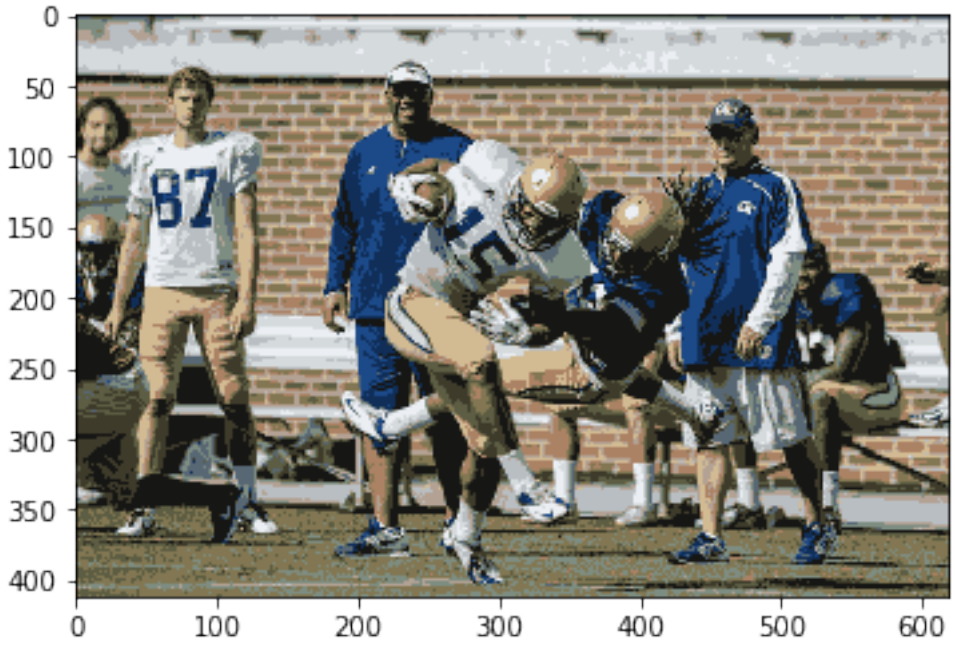
[107]:
```
lab, cen = runKMedoidsAlgorithm("data/football.bmp",4)
```

```
Running k-medoids algorithm…
iteration 1 WCSS =  795086474.0
iteration 2 WCSS =  653969572.0
iteration 3 WCSS =  627066380.0
Time taken = 380.82 seconds
Image with k = 4 clusters…
The labels are: [0 0 0 … 1 1 1]
The cluster centers are {0: array([ 29.59608288,  65.62690888, 112.67482259]),
1: array([129.45329423, 121.12920635,  95.33023646]), 2: array([198.27617676,
192.57896966, 184.46228852]), 3: array([41.51326068, 40.64565853, 32.77022857])}
```



[108]:
```
lab, cen = runKMedoidsAlgorithm("data/football.bmp",14)
```

```
Running k-medoids algorithm…
iteration 1 WCSS =  300888819.0
iteration 2 WCSS =  177207394.0
iteration 3 WCSS =  171816200.0
Time taken = 153.89 seconds
Image with k = 14 clusters…
The labels are: [0 0 0 … 8 8 8]
The cluster centers are {0: array([92.10732696, 93.67772951, 67.18986389]), 1:
array([172.23467107, 130.04667873, 101.3762508 ]), 2: array([ 88.55267126,
103.30450237, 116.30116329]), 3: array([210.25700383, 187.12112946,
147.51408139]), 4: array([213.62941392, 218.83488779, 221.56828086]), 5:
array([185.54045959, 190.50844043, 194.81077107]), 6: array([179.07417878,
162.7141947 , 148.90364324]), 7: array([235.88742656, 239.81786134,
243.79059929]), 8: array([130.36939915, 119.26123379,  82.05742072]), 9:
array([22.00743449, 22.11973768, 20.50276527]), 10: array([16.08882979,
46.85638298, 88.30159574]), 11: array([ 28.08408743,  73.822776  ,
135.71098427]), 12: array([62.60444485, 59.38514462, 43.01787897]), 13:
array([126.16531975, 141.5979152 , 129.76446475])}
```



[109]:
```
lab, cen = runKMedoidsAlgorithm("data/football.bmp",50)
```

```
Running k-medoids algorithm…
iteration 1 WCSS =  105560989.0
iteration 2 WCSS =  66334801.0
iteration 3 WCSS =  60459670.0
iteration 4 WCSS =  57889609.0
Time taken = 130.47 seconds
Image with k = 50 clusters…
The labels are: [ 6 35 35 … 43 43 43]
The cluster centers are {0: array([132.31554054, 148.33006757, 175.23108108]),
1: array([180.16598192, 169.34552177,  97.80073952]), 2: array([210.48488603,
214.78580278, 214.3327552 ]), 3: array([169.98135755, 186.3001912 , 223.1042065
]), 4: array([167.67774011, 176.62079096, 182.03954802]), 5:
array([201.16790831, 175.97191977, 161.0913085 ]), 6: array([ 87.305041  ,
99.62131795, 107.39386578]), 7: array([230.83074451, 217.69523299,
143.39314408]), 8: array([252.68463612, 252.23315364, 248.58490566]), 9:
array([42.55396766, 47.48401655, 46.44697255]), 10: array([183.07935504,
192.170724  , 198.70882074]), 11: array([23.73424271, 26.4474757 ,
23.37284415]), 12: array([124.17273559, 128.51125343, 129.92369625]), 13:
array([ 16.24119672,  53.33545142, 102.35901509]), 14: array([181.97880006,
160.04752667, 145.29333518]), 15: array([134.33531915, 160.0470922 ,
108.10695035]), 16: array([229.22205551, 233.73543386, 237.02175544]), 17:
array([197.50097213, 187.7433571 , 119.13480233]), 18: array([161.55821213,
103.100352  ,  79.94979079]), 19: array([ 7.33829224,  9.51609945,
12.02761361]), 20: array([154.90515873, 157.31626984, 146.61646825]), 21:
```

array([245.70860566, 248.66557734, 252.03213508]), 22: array([210.85182768,
222.4575718 , 243.97845953]), 23: array([186.30210773, 173.65858147,
161.66711275]), 24: array([216.01164144, 201.99592549, 120.56111758]), 25:
array([43.48606688, 73.02707006, 120.28264331]), 26: array([46.40448962,
86.74544684, 153.87462939]), 27: array([100.43980222, 121.92830655,
151.32682324]), 28: array([62.68467078, 54.32896091, 29.38271605]), 29:
array([73.79702048, 69.19180633, 50.62876547]), 30: array([180.21917627,
136.01946762, 114.9188187]), 31: array([194.78248281, 189.1514658 ,
179.87513572]), 32: array([192.21796835, 205.91985707, 231.78815722]), 33:
array([155.58940043, 126.64864383, 108.10331906]), 34: array([11.81457663,
38.13210075, 75.8204716]), 35: array([60.21021963, 70.28955625, 75.22052891]),
36: array([10.24493164, 23.17586044, 45.79561528]), 37: array([198.62847409,
204.15545992, 204.56051641]), 38: array([221.56443935, 225.27626939,
224.60331453]), 39: array([210.67559944, 199.67207334, 185.29830748]), 40:
array([6.25477154, 65.43175246, 129.51908618]), 41: array([92.08281999,
95.75952544, 50.52589551]), 42: array([234.69634703, 239.95091324,
248.67522831]), 43: array([119.98592383, 143.43083602, 87.98191146]), 44:
array([43.49216783, 36.86587413, 21.60965035]), 45: array([228.13007457, 179.
, 160.24440762]), 46: array([196.59147916, 156.12098857, 136.65805976]), 47:
array([113.12531306, 116.91878354, 79.859839]), 48: array([99.49529988,
87.2148815 , 72.15305177]), 49: array([27.02577622, 17.34094903, 8.72075766])}