INTRODUCTION TO ARTIFICIAL INTELLIGENCE

LAB ASSIGNMENT # 12

Adaptive Smoothing and HAC

PREPARED BY:

Group No. 4
Sandhya Yadav-201652023
Payal Mahisaniya-201652015
Pooja Gurjar-201652016

PROBLEM STATEMENT 1:

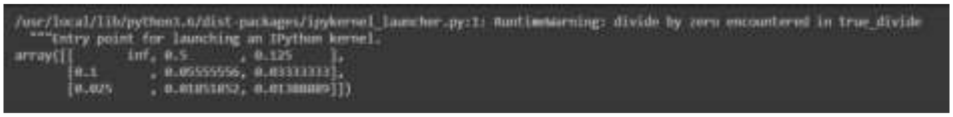
Given the noisy image test_noisy.jpg (available in the laboratory_work folder). Use the adaptive smoothing (Ref. Diffusions and Confusions in Signal and Image Processing) process to remove the noise. Explain the selected values of the parameters in the algorithm.

Python Code:

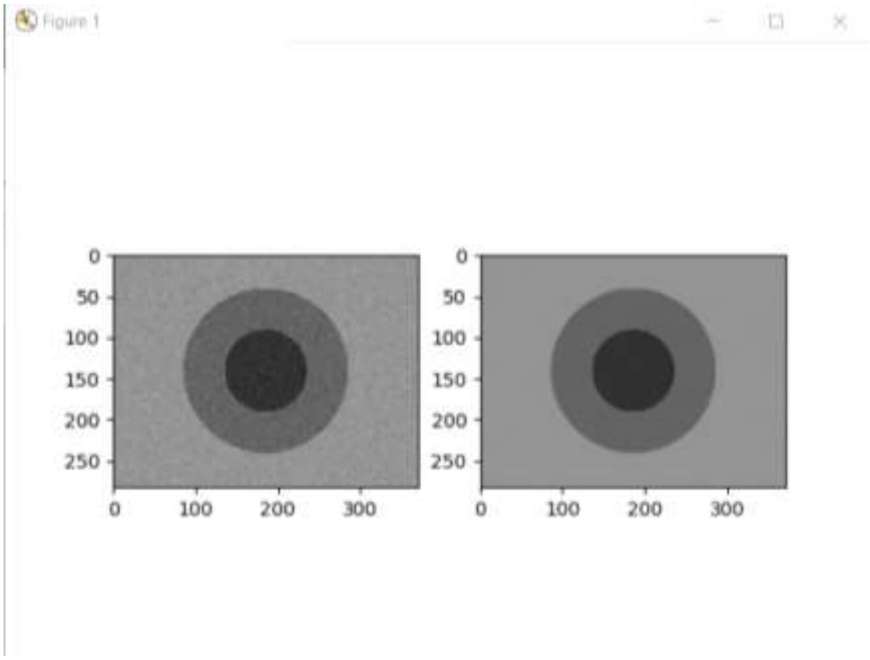
Commented [SY1]:

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

def dist(i, j, k, l, kernel):
   return np.square(i - k) +  np.square(j - l) + np.square(kernel[i, j] -  kernel[k, l])

def get_sum_beta(i, j, dist_mat, gamma):
    m = dist_mat.shape[0]
    n = dist_mat.shape[1]
    s =0
    for k in range(m):
       for l in range(n):
           if i == k and l == j:
                s +=1 / (1 + np.power(dist_mat[i,j,k,l], gamma))
    return s

def get_beta(i, j , alpha, dist_mat, gamma):
    dist_sum = alpha / get_sum_beta(i, j, dist_mat,  gamma)
    return dist_sum

def get_wt(i, j, k, l, alpha, dist_mat, kernel):
    return get_beta(i, j, alpha, dist_mat, kernel) / (1 + np.power(dist_mat[i, j, k, l], gamma))

def create_distance_matrix(kernel):
    m,n = kernel.shape
    dist_mat = np.zeros((m,n,m,n))
    for i in range(m):
        for j in range(n):
            for k in range(m):
                for l in range( n):
                    dist_mat[i, j, k, l] = dist(i, j, k, l, kernel)
    return dist_mat

def create_weight_matrix(kernel, alpha, dist_mat):
    m,n = kernel.shape
    weight_mat = np.zeros((m,n,m,n))
    for i in range(m):
        for j in range(n):
            for k in range(m):
                for l in range( n):
                    weight_mat[i, j, k, l] = get_wt(i, j, k, l, alpha, dist_mat, kernel)
    return weight_mat

def adaptive_smoothing_images(kernel, alpha=0.2):
    dist_matrix = create_distance_matrix(kernel)
    weight_mat = create_weight_matrix(kernel, alpha, dist_mat)
    pass


kernel = np.array([[1,2,3],[4,5,6], [7,8,9]])
d = create_distance_matrix(kernel)
print(1 / d[0,0,:, :])



# Using cv2

img = cv2.imread('test_noisy.jpg')

dst = cv2.fastNlMeansDenoisingColored(img,None,10,10,7,21


plt.subplot(121),plt.imshow(img)
plt.subplot(122),plt.imshow(dst)
plt.show()
```

Output:



```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
array([[       inf, 0.5       , 0.125     ],
       [0.1       , 0.05555556, 0.03333333],
       [0.025     , 0.01851852, 0.01388889]])
```



PROBLEM STATEMENT 2:

Use HAC with Euclidean/ Manhattan distance as a measure (Single link, complete link, Ward's distance, Group average, Centroid, Clusteroid) cluster the states of India based on the feature vector comprising of the following parameters (for one of the financial year values available in the data-set)

Percentage of schools with electricity
Percentage of schools with girls toilet
Percentage of schools with drinking water
Percentage of schools with boys toilet

For second problem you should get the data from the following:

https://data.gov.in/

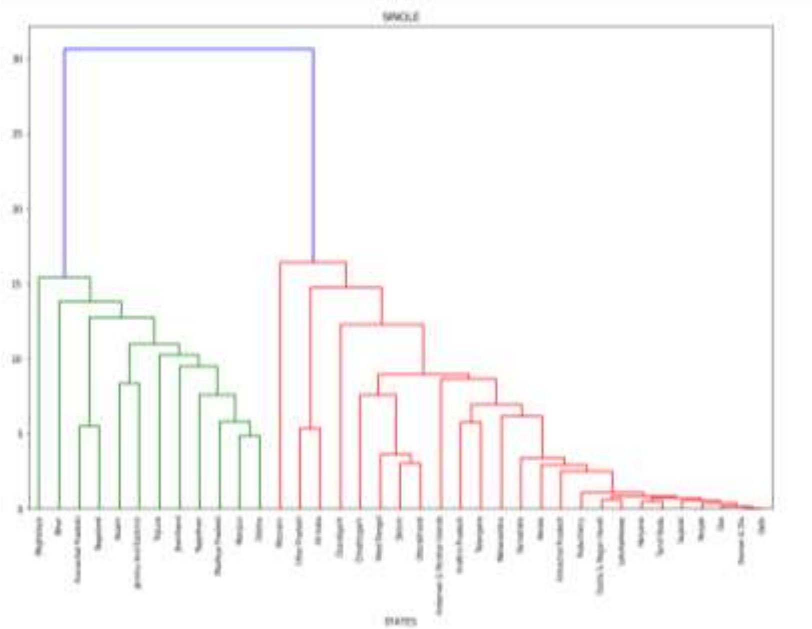
As an example, a simple visualization based on percentage of schools with electricity is available at https://data.gov.in/major-indicator/percentage-schools-electricity

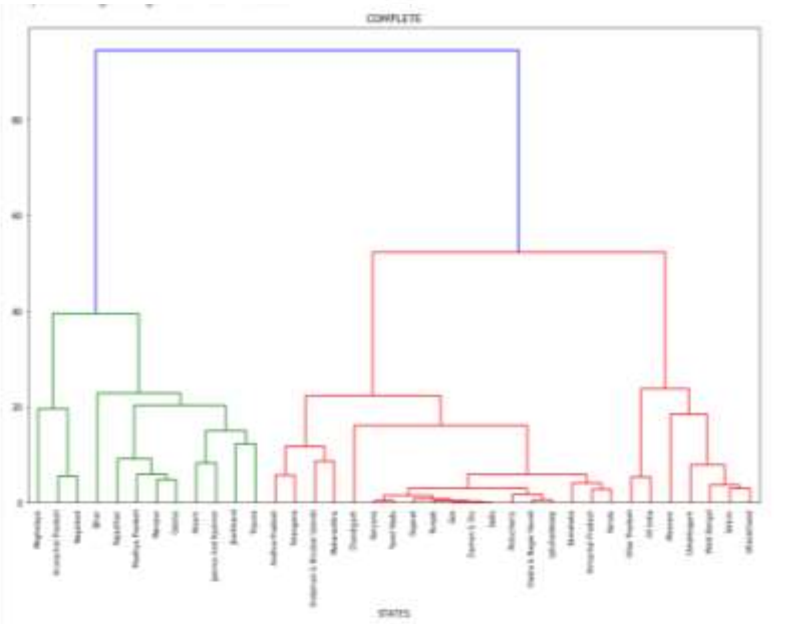
Code:

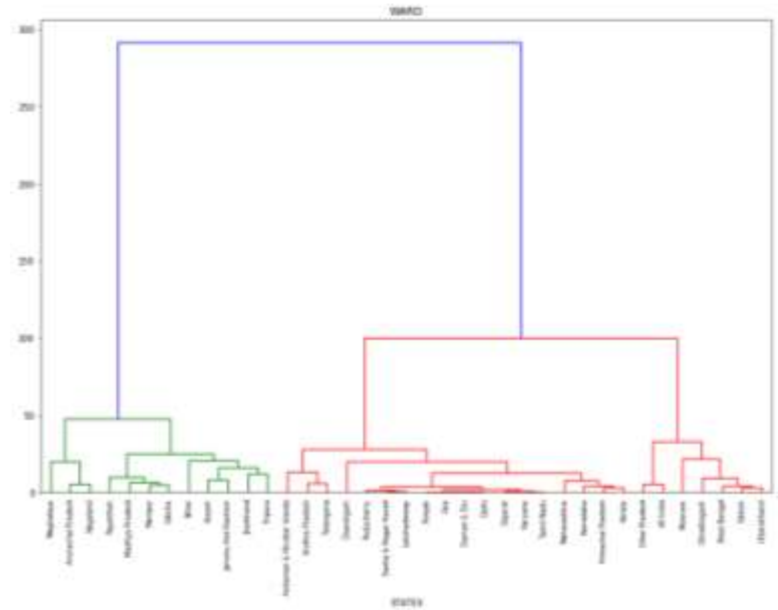
```
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt
```

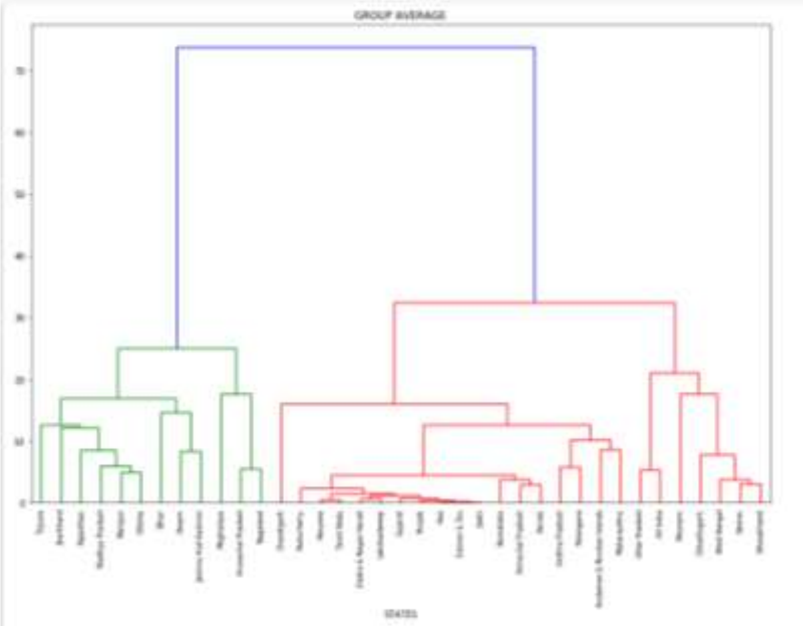
```
data = pd.read_csv('data.csv')
print(data.columns)

measures = ['single', 'complete','ward','average','centroid','median']
measureTitles = ['SINGLE', 'COMPLETE','WARD','GROUP
AVERAGE','CENTROID','CLUSTEROID']

attribute=data.columns[1:]
for i in range(len(measures)):

    ytdist = data.set_index('State')[attribute]
    Z = hierarchy.linkage(ytdist, measures[i])

    labelList = data['State']
    fig, axes = plt.subplots(figsize=(15,10))
    dn =
hierarchy.dendrogram(Z,orientation='top',ax=axes,distance_sort='descending',show_lea
f_counts=False)
    labels = [data['State'][i] for i in dn['leaves']]
    axes.set_xticklabels(labels, rotation='vertical')
    plt.xlabel('STATES')
    plt.title(measureTitles[i])
    plt.show()
    plt.figure()
```

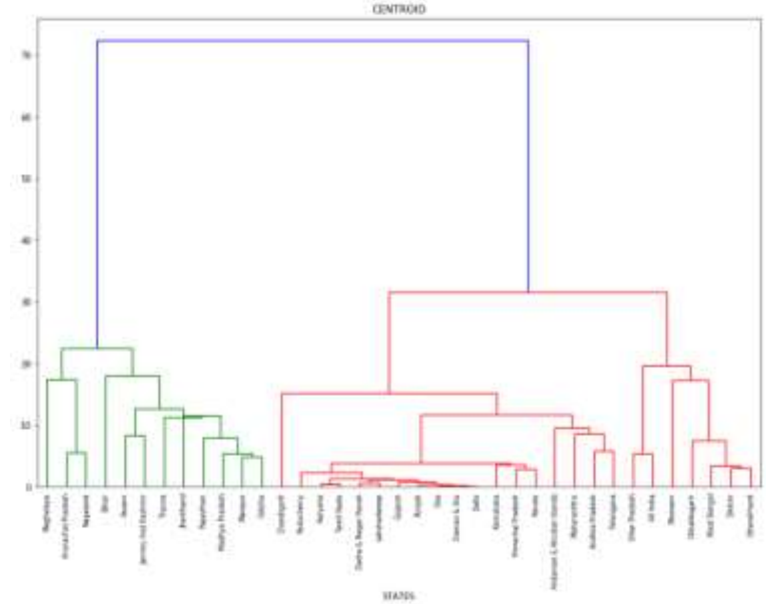
Outputs:

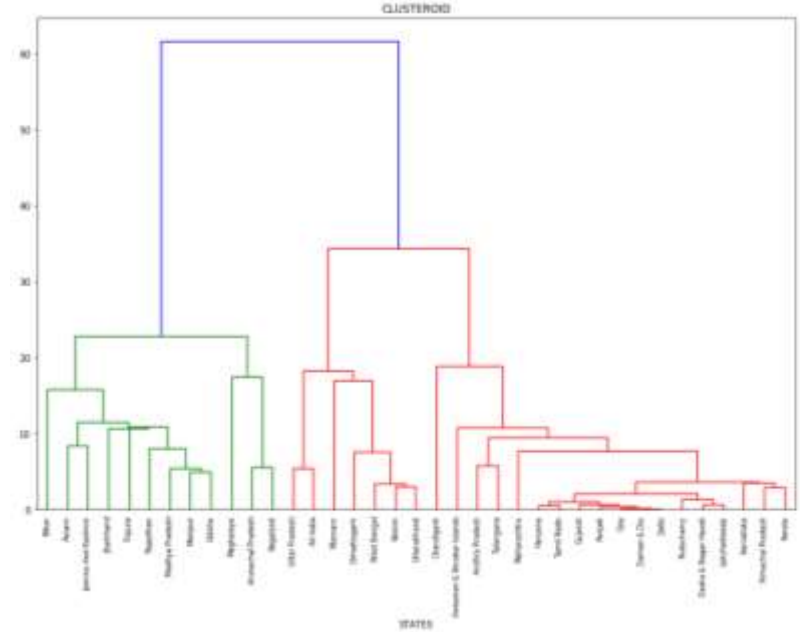

SINGLE
30
25
20
15
10
5
0
STATES


COMPLETE
80
60
40
20
0
STATES


WARD
0
50
100
150
200
250
300
STATES


GROUP AVERAGE
0
10
20
30
40
50
60
70
STATES


CENTROID
0
10
20
30
40
50
60
70
Meghalaya
Arunachal Pradesh
Nagaland
Bihar
Assam
Jammu And Kashmir
Tripura
Jharkhand
Rajasthan
Madhya Pradesh
Manipur
Odisha
Chandigarh
Puducherry
Haryana
Tamil Nadu
Dadra & Nagar Haveli
Lakshadweep
Gujarat
Punjab
Goa
Daman & Diu
Delhi
Karnataka
Himachal Pradesh
Kerala
Andaman & Nicobar Islands
Maharashtra
Andhra Pradesh
Telangana
Uttar Pradesh
All India
Mizoram
Chhattisgarh
West Bengal
Sikkim
Uttarakhand
STATES


CLUSTEROID
0
10
20
30
40
50
60
Bihar
Assam
Jammu And Kashmir
Jharkhand
Tripura
Rajasthan
Madhya Pradesh
Manipur
Odisha
Meghalaya
Arunachal Pradesh
Nagaland
Uttar Pradesh
All India
Mizoram
Chhattisgarh
West Bengal
Sikkim
Uttarakhand
Chandigarh
Andaman & Nicobar Islands
Andhra Pradesh
Telangana
Maharashtra
Haryana
Tamil Nadu
Gujarat
Punjab
Goa
Daman & Diu
Delhi
Puducherry
Dadra & Nagar Haveli
Lakshadweep
Karnataka
Himachal Pradesh
Kerala
STATES