

Project Report

Pattern Discovery In Textures

Name:Keshav Anand

Course: AI and ML

(Batch-4)

Duration: 12 months

Problem Statement: Pattern discovery in textures using Spherical K-means.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url <https://www.python.org/downloads/> can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: <https://www.pythongcentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/>. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/> You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy

Dataset used:

The dataset used is YALE Face dataset which is available on a website called Kaggle.com.

Method used for Process:

Spherical K-MEANS

Screenshots of Source Code and Output:

In [3]:

```
import os
import numpy as np
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from sklearn.feature_extraction.image import extract_patches_2d
```

In [4]:

```
root = os.getcwd()
train_folder = os.path.join(root, "yalefaces")
train_files = os.listdir(train_folder)
for i in range(len(train_files)):
    file = os.path.join(train_folder, train_files[i])
    image = mpimg.imread(file)
    plt.imshow(image)
    plt.xticks([])
    plt.yticks([])
    plt.show()
```

The notebook displays two images. The top image is a color photograph of a man's face, and the bottom image is a heatmap representation of the same face, likely showing the results of feature extraction or clustering.

In []:

```
data_arr = []
root = os.getcwd()
train_folder = os.path.join(root, "yalefaces")
train_files = os.listdir(train_folder)
for i in range(len(train_files)):
    file = os.path.join(train_folder)
    image = mpimg.imread(file)
    image_patches = extract_patches_2d(image, (8,8), max_patches = 200)
    for j in range(len(image_patches)):
        patch = np.ravel(image_patches[j])
        data_arr.append(patch)

data_arr = np.matrix(data_arr)
print(data_arr.shape)
```

In []:

```
K = 3
label_arr = np.zeros(data_arr.shape[0]), dtype = "int32"
for i in range(label_arr.shape[0]):
    label_arr[i] = np.random.choice(K)
```

In []:

```
def similarity(vec1, vec2):
    vec1 = np.ravel(vec1)
    vec2 = vec1/np.linalg.norm(vec1)

    vec2 = np.ravel(vec2)
    vec2 = vec2/np.linalg.norm(vec2)

    angle = np.dot(vec1, vec2)
    return angle
```

Home Page - Select or create a ... SPHERICAL K-MEANS PROJECT

localhost:8888/notebooks/SPHERICAL%20K-MEANS%20PROJECT.ipynb

jupyter SPHERICAL K-MEANS PROJECT Last Checkpoint 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
vec2 = np.ravel(vec)
vec2 = vec2/np.linalg.norm(vec2)

angle = np.dot(vec1, vec2)
return angle

In [ ]: def init_centroids(K, data_arr, label_arr):
    mean_cent = []
    size_cent = []
    cluster_cent = [[ ] for i in range(K)]

    for i in range(len(data_arr)):
        for k in range(K):
            if label_arr[i] == k:
                data_pt = np.ravel(data_arr[i, :])/np.linalg.norm(np.ravel(data_arr[i, :]))

    for k in range(K):
        cluster_mat = np.matrix(cluster_cen[k])
        pointnum = cluster_mat.shape[0]
        mean_k = np.mean(cluster_mat, axis = 0)
        mean_k = np.ravel(mean_k)/np.linalg.norm(np.ravel(mean_k))
        mean_cen.append(mean_k)
        size_cen.append(pointnum)
    return mean_cen, size_cen

In [ ]: def label_update(prev_mean, data_arr, label_arr):
    for i in range(len(data_arr)):
        sim_pt = []
        for k in range(K):
            sim = similarity(data_arr[i], prev_mean[k])
            sim_pt.append(sim)
        sim_arr = np.array(sim_pt)
        new_label = np.argmax(sim_arr)
        label_arr[i] = new_label
    return label_arr
```

Home Page - Select or create a ... SPHERICAL K-MEANS PROJECT

localhost:8888/notebooks/SPHERICAL%20K-MEANS%20PROJECT.ipynb

jupyter SPHERICAL K-MEANS PROJECT Last Checkpoint 5 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
In [ ]: def update_cluster(K, prev_mean, prev_size, data_arr, label_arr):
    cluster_pts = [[ ] for k in range(K)]

    for i in range(data_arr.shape[0]):
        center_pt = label_arr[i]
        data_pt = np.ravel(data_arr[i, :])/np.linalg.norm(np.ravel(data_arr[i, :]))
        cluster_pts[int(center_pt)].append(data_pt)
    for k in range(K):
        if len(cluster_pts[k]) <= 0:
            cluster_mat = np.matrix(cluster_pts[k])
            pointnum = cluster_mat.shape[0]
            mean_k = np.mean(cluster_mat, axis = 0)
            mean_k = np.ravel(mean_k)/np.linalg.norm(np.ravel(mean_k))

            prev_mean[k] = mean_k
            prev_size[k] = pointnum
            new_mean = prev_mean
            new_size = prev_size
    return new_mean, new_size

In [ ]: def Spherical_Kmeans(data_arr, label_arr, maxIter):
    prev_mean, prev_size = init_centroids(K, data_arr, label_arr)

    for iter in range(maxIter):
        new_label = label_update(prev_mean, data_arr, label_arr)
        new_mean, new_size = update_centroids(K, prev_mean, prev_size, data_arr, label_arr)
        label_arr = new_label
        prev_mean = new_mean
        prev_size = new_size
        print("Iteration : {iter} is completed")
    return new_mean, new_size

In [ ]: mean_cen, size_cen = Spherical_Kmeans(data_arr, label_arr, 45)
mean_cen = np.array(mean_cen)
```