Assignment Project Report

Hashing: Querying in Face Datasets

Name: Pavan Tiwari

Course: AI and ML
(Batch 4)

- **Problem Statement**

Perform activity recognition on the dataset using a hidden markov model. Then perform the same task using a different classification algorithm (logistic regression/decision tree) of your choice and compare the performance of the two algorithms

- **Prerequisites**

  - Software:
    - Python 3 (Use anaconda as your python distributor as well)

  - Tools:
    - Pandas
    - Numpy
    - Matplotlib
    - Sklearn
    - Seaborn

  - Dataset Link: Yale Face Dataset

- **Method Used**

  Implement a basic hashing model from scratch that hashes the images. You can use any dataset of few images and can implement a-hash or any other hashing algorithm of your choice. For a-hash, given any images, first resize the image to a suitable size, followed by grayscale conversion of the image. Then mean normalize the image to obtain a binary image, whose sum can be used as a hash value. Using the hash model, encode all the images present inside your directory and then search for images similar to the query image

Load all required libraries

In [1]:
```
import cv2
import pandas as pd
import numpy as np
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
```

In [2]:
```
# Reading multiple images from a folder and storing it in a list.
folder = "./YALE"
images = []
for file in os.listdir(folder):
    img = mpimg.imread(os.path.join(folder, file))
    if img is not None:
        images.append(img)
print(images)
```

---

1. Generating Random Hash

In [4]:
```
def genRandomHashVectors(m, length):  # Generate random unit vectors for Hashing
    hash_vector = []
    for i in range(m):
        v = np.random.uniform(-1,1,length)
        vcap = v / np.linalg.norm(v)
        hash_vector.append(vcap)
    return hash_vector
```

In [13]:
```
def ahash(hash_vector ,data):
    hash_code = []
    for i in range(len(hash_vector)):
        if np.dot(data,hash_vector[i]) > 0:
            hash_code.append('1')
        else:
            hash_code.append('0')
    return ''.join(hash_code)
```

In [15]:
```
# Creating a Image Dictionary using the hash as the keys
img_dict = {}
for i in range(len(image_vector)):
    hash_code = ahash(hash_vector,image_vector[i])
    if hash_code not in img_dict.keys():
        img_dict[hash_code] = [i]
    else:
        img_dict[hash_code].append(i)
```

In [16]:
```
keys = list(img_dict.keys())
values = list(img_dict.values())
```

2. Plotting Image Function

In [18]:
```
# Plotting images with same hash code
def plotImages(images, img_indices):
    imgs = [images[i] for i in range(len(images)) if i in img_indices]
    fig = plt.figure()
    cols = 2
    n_images = len(imgs)
    for n,image in zip(range(n_images),imgs):
        ax = fig. add_subplot(cols, np. ceil(n_images/float(cols)), n + 1)
        plt.gray()
        plt.imshow(image)
    fig. set_size_inches(np. array(fig. get_size_inches()) * n_images)
    plt.show()
```

3) Generating the Hash Vector

In [3]:
```
# Vectorizing the images and storing it in a list
image_vector = []
for image in images:
    row,col = image.shape
    img_vec = image.reshape(row*col)
    img_vec_norm = img_vec / np.linalg.norm(img_vec)  # Converting the image vector to a unit vector
    image_vector.append(img_vec_norm)
print(img_vec.shape)
print(len(image_vector))
```

```
(45045,)
165
```

In [4]:
```
def genRandomHashVectors(m, length):  # Generate random unit vectors for Hashing
    hash_vector = []
    for i in range(m):
        v = np.random.uniform(-1,1,length)
        vcap = v / np.linalg.norm(v)
        hash_vector.append(vcap)
    return hash_vector
```

4) Output