Capstone Project Report

Name: Sumit Patel

Course: AI and ML (Batch – AUG 2020)

Duration: 10 months

Hashing: Querying in Face Datasets

Problem Statement:

Factor analysis is a useful technique to find latent factors that can potentially describe multiple attributes, which is sometimes very useful for dimensionality reduction. Use the Airline Passenger Satisfaction dataset to perform factor analysis. (Use only the columns that represent the ratings given by the passengers, only 14 columns). Choose the best features possible that helps in dimensionality reduction, without much loss in information.

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.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/. 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 numpy

pip install matplotlib

pip install pandas

If you have chosen to install anaconda then run below commands in anaconda prompt to install these packages:

conda install -c anaconda numpy

conda install -c anaconda matplotlib

conda install -c anaconda pandas

Dataset used:

YALE dataset

Importing the libraries and loading dataset.

```
# Importing the required libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

Local Sensitive Hashing for the images

Locatily sensitive Hashing - Random Projections

```
# Generator function to generate random unit vectors for Hashing
def genRandomHashVec(m, length):
    hash_vec = []
    for i in range(m):
        v = np.random.uniform(-1,1,length)
        v_ = v/np.linalg.norm(v)
        hash_vec.append(v_)
    return hash_vec
```

```
# Function for local sensitive hashing
def lsh(hash_vec, data_pt):
    hash_code = []
    for i in range(len(hash_vec)):
        if np.dot(data_pt, hash_vec[i])>0:
            hash_code.append('1')
        else:
            hash_code.append('0')
    return hash_code
```

```
# Generate 10 random vectors of the same size as image vector
hash_vector = genRandomHashVec(10, len(imgs_vec[0]))
np.array(hash_vector).shape
```

Testing the hashed images

```
#Test the lsh fuction
lsh(hash_vector, imgs_vec[0])
```

```
['0', '1', '0', '1', '1', '1', '0', '0', '1', '1']
```

```
# Creating an image dictionary using the hash as keys
image_dict = {}
for i in range(len(imgs_vec)):
    hash_code = lsh(hash_vector, imgs_vec[i])
    str_hash_code = ''.join(hash_code)
    if str_hash_code not in image_dict.keys():
        image_dict[str_hash_code] = [i]
    else:
        image_dict[str_hash_code].append(i)
```

```
# Display the hashes
col_names = ['Hash_Code', 'Image_Index']
df = pd.DataFrame(image_dict.items(), columns = col_names)
df.head()
```

	Hash_Code	Image_Index
0	0101110011	[0, 15, 16, 19, 21, 22, 58, 133, 134, 136, 154]
1	0111110001	[1, 2, 4, 5, 7, 8, 9, 10, 26, 27, 29, 34, 35, ...
2	0111010011	[3, 44, 47, 57, 63, 80, 91, 102, 104, 106, 109...
3	0111101000	[6, 28, 160]
4	0101111000	[11, 24, 74]

```
# Get the keys and values of the of the dictionary
keys = list(image_dict.keys())
values = list(image_dict.values())
```

```
# Plotting images with same hash cod
igs = [images[i] for i in range(len(images)) if i in values[1]]
fig = plt.figure()
cols = 2
n_images = len(igs)
for n, image in zip(range(n_images), igs):
    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()
```