Final report

Kangdong Yuan

Table of Contents:

Problem Statement

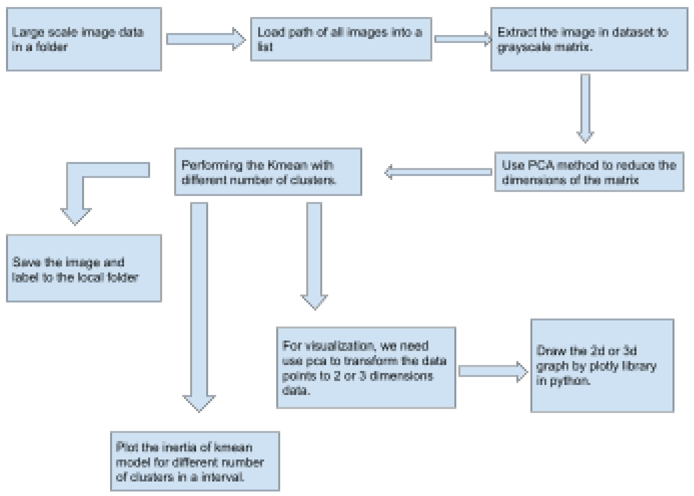
In this plan, in order to complete customer needs and make effective procedures. We decided to use machine learning to analyze and model Egyptian hieroglyphics data. We hope that through our work, we can help Professor Redford to better process the data he has. So we decided to build a supervised learning model and an unsupervised learning model. My job is to build an unsupervised model of Egyptian hieroglyphic data. In addition, I need to do a lot of work to check the accuracy of my result data, and analyze the images of each cluster to improve some clusters with lower purity.

Data

It is difficult for me to find a mature Egyptian hieroglyphic picture database. Although I can search a large number of Egyptian hieroglyphic pictures, these pictures have different colors and resolutions. Then I found a database that I can use in a GitHub code repository. In this amount of data, there are more than 4000 pictures of hieroglyphic letters, all of which are black and white pictures and have the same resolution. This database is very suitable for me to build an unsupervised learning model.
https://github.com/yedkk/kmean-model/tree/main/data

Methods

I have a total of six steps to train the unsupervised model of the picture:


Large scale image data in a folder
Load path of all images into a list
Extract the image in dataset to grayscale matrix.
Use PCA method to reduce the dimensions of the matrix
Performing the Kmean with different number of clusters.
Save the image and label to the local folder
For visualization, we need use pca to transform the data points to 2 or 3 dimensions data.
Draw the 2d or 3d graph by plotly library in python.
Plot the inertia of kmean model for different number of clusters in a interval.

First, I put more than 4000 images to be processed in a folder, and I created a result storage folder to store the clustered images.

Second, through a lot of information search, I learned that if I want to extract the features of each image, I need to convert the different pixels of each image into grayscale numbers. Then I need to convert each picture into a matrix of gray values. Then I extracted the features of each image from these matrices, and added these features to a list purely for deep learning training.

Third, when I was training my k-means model, I found that my training speed was particularly slow, I needed to find a more efficient way to train my model. Through my exploration of the matrix, I found that the matrix of each image has 224 numbers. If I use such a large matrix to train my k-means model, the cpu will face tremendous pressure and even cause the temperature to be too high and unstable. So I used the PCA method to reduce the size of the matrix. PCA is a mathematical method to convert data points in high dimensionality into data points in low dimensionality. I use the PCA library of python to call the PCA function. When I reduced the matrix dimension of the training data to 10, the training speed of my k-means model dropped from the half hour to 20 seconds. Through the PCA method, I improved the efficiency of model training to a satisfactory level.

Fourth, after all the image features were extracted, I started to use kmean to train my deep learning model with a number of clusters equal to 27.

Then, when I need to verify the accuracy of my k-means model, in addition to verifying by viewing the results, we can also verify by visualizing the k-means model. First, I extract the labels of each data point after applying the k-means model. Then in order to show the data points in plane and three-dimensional coordinates, I used the PCA method to convert ten-dimensional data points into two-dimensional and three-dimensional data points. Then I use the two-dimensional and three-dimensional data as the spatial coordinates of each data point to draw the graph. Finally, for each data point in a different cluster, I use a unique color to represent it.

Finally, I hope to obtain the accurate value of my k-means model, not only through visual image observation, but to quantify the accuracy of the k-means model digitally.

However, after checking my result data, I found that my result data is more accurate for simple geometric images, such as horizontal lines, vertical lines, semicircles, and bird shapes. But for more complex images, such as rabbits, lions, crocodiles and other images, the cluster purity is very low. I think my k-means model will have such a result because the parameters I set do not fit my database well. I search the resources and find that there are two parameters that I can change to improve my purity.

The first parameter is n_init, which is the rounds of random initialization. In my original model, I set my *n_init* parameter equal to 10, so it will choose initial centroids 10 times and return the best initialization. n_init = 10 is enough for a module with small cluster numbers, but I have 27 clusters in my model. So, I increase the n_init and find that 35 is a good choice for this parameter. Because when n_init is equal to 35, my result data can better remove some misclassified pictures in the cluster. Some clusters containing circular images and clusters containing bird images have achieved high purity. For example, my cluster 6 has achieved 0.96 purity.

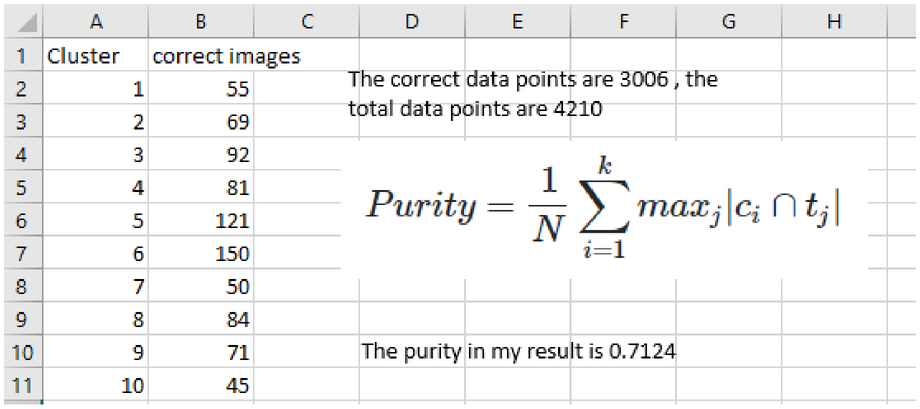
Second parameter is tol, which is the relative tolerance with regards to Frobenius norm of the difference in the cluster centers. The default tol is 1e-4, but I think I can make it smaller to enable my kmean model clustering image with more strict rules. So, I decrease the tol to 0.5e^-4 and 1e^-5. After changing the tol, I rerun my code, I find that it gives me a little improvement in my result data. For example, in clusters, there is only one misclassified image, so the purity has been improved from 0.83 to 0.99.

But for some clusters with low purity such as cluster 12, the adjustment of these parameters did not achieve good results. Although the purity has increased, it still does not satisfy me.

Experiment

In order to get the accuracy of my k-means model, I need to check each of my clustered pictures and determine how many misclassified pictures there are. But instead of doing the check to every picture. I need to find a way to calculate an accurate number to indicate the accuracy of my k-means. I searched a lot of information. I found that the best parameter to measure clustering data is purity. Within the context of cluster analysis, Purity is an external evaluation criterion of cluster quality. It is the percent of the total number of data points that were classified correctly, in the unit range 0 to 1, which 1 means no misclassified data points. Then I checked the data of each of my clusters to find out how many accurately classified data points are in each cluster. And record these data points in my excel file for summation. Then, I used a formula to calculate the purity of my result data, and found that the purity of my data was 0.7124, which is not a very good score. This means that I need to do more to make my unsupervised model more accurate.

My calculation documentation:
https://github.com/yedkk/kmean-model/blob/main/calculate%20purity.xlsx



	A	B	C	D	E	F	G	H
1	Cluster	correct images						
2	1	55		The correct data points are 3006 , the				
3	2	69		total data points are 4210				
4	3	92						
5	4	81						
6	5	121						
7	6	150						
8	7	50						
9	8	84						
10	9	71		The purity in my result is 0.7124				
11	10	45						

$$Purity = \frac{1}{N}\sum_{i=1}^{k} max_j |c_i \cap t_j|$$

When checking my result data, I found that there is a huge difference in the proportion of misclassified data between different clusters. Then I started to carefully check the accuracy of my result data, I hope to find out which clusters I have achieved better results, and in which clusters the accuracy is low. I started to analyze the clustering of which main pictures are lines. For example, in cluster 0, the main picture in this cluster is two horizontal lines, which is the letter TH in the hieroglyph. I found that there are only 8 misclassified images in this cluster, and the purity of this cluster is 0.891, which is a good result. In clusters 1, 2, 4, and 5, which are similar to clusters 0, the main images of the clusters are all one or two horizontal lines, which are the letter N, TH and H in hieroglyph. These clusters have achieved good results because there are few misclassified images in these clusters. So I assume that my k-means model has good accuracy when classifying images with lines.

Next, I analyze the clusters with birds in the pictures. For example, the main pictures in cluster 6 are the birds with the head facing to the right, which is the letter A in the hieroglyph. In this cluster, there are only three misclassified pictures. From this, I can calculate the purity of this cluster to be 0.96, which is a very good score, which means that my k-means model has succeeded in the classification letter A. In other clusters that are similar to clusters, such as clusters 9,14,19,21,22. My k-means model has also achieved good classification results. For example, the purity of my cluster 20 is 1, which means that this cluster is all accurately classified pictures.

Then, I try to analyze the clustering of which main pictures are vertical lines. These pictures are vertical knives, representing the I and Y letters in the hieroglyph. I found that my k-means model is also more accurate for the clustering of vertical line pictures. For example, in the 464 pictures in cluster 20, I got a purity of 0.73. In the 320 pictures in cluster 18, I got a purity of 0.765.

Then I analyzed the clusters where my main pictures are semicircular, including clusters 22 and 26. These pictures are the K and T letters in the hieroglyphs. Among 128 pictures in cluster 22, the purity is 0.98. The purity of 265 images in cluster 26 is 0.69. The purity of these clusters has reached a satisfactory level.

But after checking my result data, I found that the pictures in some clusters are irregular and not similar. For example, in cluster 12, there are pictures of circles, arches, rabbits, crocodiles, and boats. These are the letters L, KA, SH, and D in the hieroglyphs. Because I couldn't find the main image in this cluster, I determined that the purity of the 192 images in the cluster is 0. I also found the same problem in cluster 16. There are many misclassified images in the cluster. The main picture of cluster 16 is a blade of grass, which represents the SU letters of hieroglyphs. But among the 172 images in cluster 16, more than half of the misclassified images, I calculated that the purity of cluster 16 is 0.37. Regarding the differences in the purity of different clusters in my result data, I think this is because my k-means model cannot do well on complex graphics, such as plants and animals. Moreover, when I see my group members use a different formula to calculate purity, I also try to use this formula to calculate my data. I found that my purity is 4.3, compared to 4.5 for her model. The purity in the new formula is also lower than Xueqing's purity.

Results

In my kmean model, the parameters I use are n-clusters = 27, n-init = 35, max-iter = 300, tol = 1e-5, random_state = 0. And, in the PCA function, I set the dimension = 20. The final purity is 0.7124, which has the space to improve.
In the output folder, each clustered picture is represented by a different file name to make it easier to check the accuracy of the result data. In addition, the running time of this program in the colab environment is within 10 minutes, and the running time on the local 8-core processor is within half an hour.
Result data： https://github.com/yedkk/kmean-model/tree/main/result

Discussion

First, the pictures in my data set are all black and white pictures. In addition, the resolution of these pictures is very low, and some of them have overexposure problems. I think if we can get high-resolution color pictures to train my model, maybe my model will produce better results. For my kmean model, I think I can make a lot of improvements. For example, I can try more parameter choices. And I think that using the k-means model to process data has inherent disadvantages, because the center of k-means is initialized randomly, so the result of the k-means model is unstable. I think if I use kmeans++ to train my model, I might get better results. And I learned through some materials, Generative Adversarial Networks (gan) has better accuracy in the field of unsupervised learning. Some gan projects have amazing accuracy in the classification of images. But since I have not learned how to use gan, I did not build my gan model successfully this semester.

Conclusion

During this semester, I worked hard to participate in the group plan and devoted my energy to training and improving my unsupervised learning model. I am more satisfied with the results of my k-means model, but I think there is still a lot to do to improve my k-means model. For example, find a better way to improve the accuracy of my k-means model to distinguish complex graphics. But I am very happy with the communication and cooperation between me and the professors and group members this semester, because I have learned a lot of important knowledge and skills from professor and group members.

Future work

In the future, I think I need to do three things to improve our group's plan. First, I need to find more training data. Although we already have 4000 pictures, we need more high-resolution pictures to help us train machine learning models.
Second, I need to try more machine learning models, because the kmans++ model is a more stable model than the k-means model. Moreover, the Generative Adversarial Networks model is a better way to classify our existing data. So trying other machine learning models may allow us to obtain higher accuracy.
Third, because our existing formulas for calculating the purity of the result data are not the same, we need to find and decide to use a better formula to calculate the purity of our result data in the future.

Reference:

Kavyazin, D. (2020, October 6). Principal Component Analysis and K-means Clustering to Visualize a High Dimensional Dataset. Medium. https://medium.com/@dmitriy.kavyazin/principal-component-analysis-and-k-means-clustering-to-visualize-a-high-dimensional-dataset-577b2a7a5fe2.

Sirait, K. (2017). K-Means Algorithm Performance Analysis With Determining The Value Of Starting Centroid With Random And KD-Tree Method. Journal of Physics: Conference Series. https://hkvalidate.perfdrive.com/?ssa=99ff7f93-657e-4ec0-90fd-f782876d1d7e&ssb=59510299099&ssc=https%3A%2F%2Fiopscience.iop.org%2Farticle%2F10.1088%2F1742-6596%2F930%2F1%2F012016%2Fpdf&ssi=8b285666-8427-4578-8d68-02f94af7afd9&ssk=support@shieldsquare.com&ssm=37926824710280114107755214454373&ssn=e11b1fe061b95c540235a6972bc9ac65ba3ebdff6237-dc5b-4dce-941133&sso=578414db-64b7bb48ea051e529513df8a8ae8cfe43e7374b7b8305b4d&ssp=04467851151619656451161962642570827&ssq=25140677067533281378170675372519440020199&ssr=ODMuOTYuMjQ1LjIyMQ==&sst=&ssv=&ssw=

savyakhosla. (2020, June 11). ML: K-means++ Algorithm. GeeksforGeeks. https://www.geeksforgeeks.org/ml-k-means-algorithm/.

Brownlee, J. (2019, July 19). A Gentle Introduction to Generative Adversarial Networks (GANs). Machine Learning Mastery. https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

Appendix:

Kmean, PCA and visualization code in jupyter notebook: https://github.com/yedkk/kmean-model

```
from keras.preprocessing import image

from keras.preprocessing.image import load_img

from keras.preprocessing.image import img_to_array

from keras.applications.vgg16 import preprocess_input

from PIL import Image as pil_image

from keras.applications.vgg16 import VGG16

from keras.models import Model

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

import os, shutil, glob, os.path

import numpy as np

import matplotlib.pyplot as plt

from random import randint

import pandas as pd

import pickle

import torch

image.LOAD_TRUNCATED_IMAGES = True

model = VGG16(weights='imagenet', include_top=False)

#input and output path

imdir = '/content/data'

targetdir = r"/content/result"

filelist = glob.glob(os.path.join(imdir, '*.png'))

# filelist.sort()

featurelist = []

########################################################################
```

#Adopted from
https://franky07724-57962.medium.com/using-keras-pre-trained-models-for-feature-extraction-in-image-clustering-a142c6cdf5b1

```
for i, imagepath in enumerate(filelist):

    img = image.load_img(imagepath, target_size=(224, 224))

    img_data = image.img_to_array(img)

    img_data = np.expand_dims(img_data, axis=0)

    img_data = preprocess_input(img_data)

    features = np.array(model.predict(img_data))

    featurelist.append(features.flatten())

#######################################################################



n_digits=10

pca = PCA(n_components=n_digits).fit_transform(featurelist)

number_clusters=27

kmeans = KMeans(n_init=35, n_clusters=number_clusters, random_state=0, tol=0.00001)

kmeans.fit(pca)

try:

    os.makedirs(targetdir)

except OSError:

    pass

for i, m in enumerate(kmeans.labels_):

    shutil.copy(filelist[i], targetdir +'\\'+ str(m) + "_" + str(i) + ".jpg")

ran = range(25, 35)

errorlist = []

for k in ran:

    kmeans = KMeans(n_init=35, n_clusters=number_clusters, random_state=0, tol=0.00001)

    kmeans.fit(pca)

    errorlist.append(kmeans.inertia)

plt.plot(ran, errorlist, '-o', color='black')
plt.xlabel('number of clusters, k')

plt.ylabel('inertia')

plt.xticks(ran)

plt.show()

n_digits=2

pca = PCA(n_components=n_digits).fit_transform(featurelist)

number_clusters=27

kmeans = KMeans(n_clusters=number_clusters, random_state=0)

kmeans.fit(pca)

label = kmeans.fit_predict(pca)

centers = kmeans.cluster_centers_

u_labels = np.unique(label)

 #####################################################################################
```

Adopted from https://www.askpython.com/python/examples/plot-k-means-clusters-python

```
for i in u_labels:

    plt.scatter(pca[label == i , 0] , pca[label == i , 1] , label = i)

plt.scatter(centers[:,0] , centers[:,1] , s = 80, color = "k")

fig = plt.figure(figsize=(60,60))

plt.show()

####################################################################################

n_digits=3

tri = PCA(n_components=n_digits).fit_transform(featurelist)

list1 = label.tolist()

list2 = [[int(item) for item in list1]]

label = np.array(list2)

d3=np.concatenate((tri, label.T), axis=1)

d3 = d3[0:4000]

label = label[0:4000]

list1 = label.tolist()
```

```
list1=label[0]

list1 = list1.tolist()

import plotly.graph_objects as go

t = np.linspace(0, 20, 100)

x, y, z = d3[:,0], d3[:,1], d3[:,2]

#########################################################################
```

Adopted from https://plotly.com/python/3d-scatter-plots/

```
fig = go.Figure(data=[go.Scatter3d(

    x=x,

    y=y,

    z=z,

    mode='markers',

    marker=dict(

        size=5,

        color=list1,                # set color to an array/list of desired values

        colorscale='Viridis',

        opacity=0.7

    )

)])

fig.update_layout(margin=dict(l=0, r=0, b=0, t=0))

fig.show()

#############################################################################
```