Hieroglyphics

Data Science 440
Project Report (to customer)

Team Six

Juliana (Jiayin) Hu, Sydney Wehn, Kangdong Yuan, Xueqing Zhang, Yuqi Gao

Table of Contents:

Problem Statement

At the beginning of the semester, our team was challenged with the task of making sense of hieroglyphics for our client Dr. Redford. During our early meetings, we found that the customer wanted us to use our data science backgrounds to help with scribe identification. As a group, we then discussed how instead, we would use supervised and unsupervised clustering and classification on the hieroglyphics dataset. For this problem, our labels are the Egyptian alphabet that corresponds to the Gardiner Sign list of these characters and their associated meanings.

Data

We used two different data sets in our project, the first dataset is the Egyptian Hieroglyphs dataset, which contains 4210 images found in the Pyramid of Unas. They are labeled with the Gardiner label by the data provider. Below I attached the link to the dataset. https://github.com/morrisfranken/glyphreader This data was also used in other Egyptian Hieroglyphics researches, which adds to the credibility of the data.

Method

K-Mean clustering

Kangdong use six steps to build the unsupervised model of the picture:

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.

Agglomerative Clustering:

As for building the model to fit the first dataset, which is the Egyptian Hieroglyphs dataset, we first applied three different feature descriptor methods. The three feature descriptor methods are Hog, Canny, and Embedding. I mainly researched the Canny method, which is to reduce the noise and look for the intensity gradient of the image. Then Juliana and Xueqing worked together to combine the work and generated a complete embedding to include all three feature descriptors and all the images we have in the data file. We first create the initial embedding with all three feature descriptors we built, but it does not run fluently at the first time. So we set up a meeting to write the following three functions including getting the number of PCAs. Last but not least, we applied these three feature descriptors to our complete data using Agglomerative Clustering. Since our team is working with image data and not a large amount of datasets, we all think

unsupervised clustering is beneficial for the practice. Two of my teammates are doing the k-means clustering, so we choose to do a different clustering to see if any difference will happen. Also, since we had worked on the feature descriptor, we decided to use all of them together. The model successfully clusters the images, and we need to evaluate the performance of our model.

Feature Descriptor: K-means

Xueqing and Jiayin searched online and decided to use feature descriptors to get the information of every picture's appearance. And then doing clustering models using the data we gathered from feature descriptors. Feature descriptor works by representing only the most important information about the image. There were lots of different kinds of feature descriptors and Xueqing used the HOG feature descriptor. Using the words from the resource she found, HOG works by extracting the information of the edges magnitude and the orientation of the edges. Xueqing worked on generating the HOG with one image first. And then applied HOG for all the images in the dataset. Jiayin did another feature descriptor called Canny. Then Xueqing and Jiayin worked together on doing another feature descriptors called inception, later they combined the data from three feature descriptors together into one file. Xueqing did the K-means clustering using the combined feature descriptors data and Jiayin worked on another clustering model called Agglomerative.

ResNet34

AlexNet

Experiment - impurity, accuracy

K-mean clustering:

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: 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 total data points are 4210				
3	2	69						
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.

Agglomerative Clustering:

Once the model is built, the next step is to find out the best parameters. I tried different clustering numbers to see what number results in the best performance. After I finished the clustering, I used the elbow method and found out the best result returned without overfitting had the number of clusters of 30. Since 27 is pretty close to 30 and the elbow method's result is not precise to 1 but 10, which means the interval returned by the elbow method is 10,20,30, and etc. Thus, I decided to examine the cluster of 27 more and also because it is also the number of Egyptian alphabets. I evaluated my cluster based on the number of labels in each cluster, and found the top 2 labels, and calculated their percentages. As a result, I found out that the Agglomerative clustering I worked on had 15 out 27 clusters containing more than 80% of the top 1 label, which means the cluster does include 1 label instead of mixing up the labels. The rest clusters that have similar percentages for top 1 and top 2 actually have similar pictures. All the characteristics indicate that it is a model that could fit our data well. But I still want to see if the model will work well on other data without being affected by the image colors or backgrounds. That's when I tried to fit the yoga data with the same clustering method. For the yoga data, I followed a similar process as I did for the Egyptian Hieroglyphics data, which fit the data in the model and tested different numbers of clusters including 30,60, and 100. Then I started to compare the results with the team.

Feature Descriptor K-means:

For my K-means model, I have first set the cluster number for 100. After talking to the professor about my work, I decided to try more different cluster numbers to find out the best one. I randomly tested with different cluster numbers as 100, 50. Then I used the elbow method to find out the best cluster number as 30. After taking the professor's advice, I chose 27 as the final cluster number since there are 27 letters in Egyptian alphabets.

Results - impurity, accuracy with best hyperparameters

As a team, we wanted to find out the best model that we build throughout the semester. First, Juliana compared her model with Xueqing, because they used similar feature descriptors and the only difference is the method they use. One used k-means and the other used Agglomerative Clustering. They both did a table to display our results, mine is shown below. The agglomerative

clustering had many clusters with only 1 image, which should mean the model has a good understanding of images and clusters them into the right place. Nevertheless, as the cluster 3 and 8 show, the image is actually the same one. This does make the model lose some accuracy, and when we later compute the Gini Impurity, the model is a bit worse than Xueqing's, which is 5.6 compared to 4.9. She was using feature extractions that are a bit different from ours. And, in kangdong's kmean model, the parameters are n-clusters = 27, n-init = 35, max-iter = 300, tol = 1e-5, random_state = 0. And, in the PCA function, kangdong set the dimension = 20. The final purity is 0.7124, which has the space to improve. And kangdong calculates his result data purity in Xueqing's formula, getting 5.3 purity. So Xueqing's model is the best among our group. Among the three of us, his clustering is the best we came up with. In addition, the teammates that did the classification for the data provided two results. One for ResNet34, one for AlexNet; the accuracy is about 69% and 59%. So between the two classification models, ResNet is better.

Discussion - personal limitations

One of the biggest limitations for our project is that we do not have a large number of dataset. 4000 images are not considered much when building a training model that could fit well on other image data. Aswe had experienced, our model fit pretty well with the Egyptian Hieroglyphics data but not good enough for the yoga data. It could probably because our original does not have color and background as noise for the feature descriptor. So we think it is a limitation we struggled with for a pretty long time.

We 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. We think if we use kmeans++ to train my model, we might get better results.
And we 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 we have not learned how to use gan, we did not build our gan model successfully this semester.

Conclusion - value to customer/ professor

For the feature descriptors, the k-means model performed well in the Egyptian Hieroglyphics dataset. It has correctly grouped most of the pictures. Which is easier for the customer to group large amounts of new pictures.

Future Work

any other image processing problems that use classification or clustering, making code transferable to future studies

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.

Cartouche Detection (Extra Content)

The cartouche pattern can be marked as a signature area or the name of the person who spoke the speech recorded on a plate in the Egyptian Hieroglyphs, it can be meaningful to have a method to quickly detect the cartouche pattern inside an image and can improve the efficiency of the researchers studying the language when looking at huge amount of records.

Using a self built dataset, from scraping from the original resources digitized PDFs on the Internet, it is able to train a model to detect the cartouche pattern.

From a total of 2,700 images scraped from the PDFs, 501 images are picked to be annotated for the purpose of training the model, using tool called LabelIMG, it is possible to annotate the images and marked the area of the object and label the area as a name the users want, and output .xml file which can be put into deep learning algorithm. After separating the dataset into 80% for training data and 20% for validation data.

Using tensorflow as the backend and imageAI as the tool to train the model, the output model can have at highest 88% certainty of a detected cartouche pattern.

The model was still not very accurate at the moment due to the dataset is relatively small, with a bigger training dataset, it is possible to improve the model to a higher accuracy and thus will provide a better probability when detecting a cartouche pattern inside an image.

References

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

https://colab.research.google.com/drive/1jqP8RmVEeXl0ATujQFmOWH8JQOImE5w7?usp=sharing (Wehn)

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

(Kangdong)

Feature descriptor, Kmeans, Agglomerative Clustering:

https://github.com/Jiayin-Hu/DS-440.git(Jiayin Hu & Xueqing Zhang)

Code used for cartouche detection:

https://github.com/Yuqi-Gao/DS440/blob/main/Code/cartouche_detection.py