Lab 4 - Automated Hieroglyphics Reading

DA350

February 16, 2018

In this lab you will build a system for reading Egyptian hieroglyphics. Specifically, you will cluster similar hieroglyphics together and devise a way of automating the assignment of new images. This can help researchers and museums interpret the meanings of these ancient messages. If for example an Egyptologist determines the meaning of each symbol, our system can quickly output the message of an arrangement of symbols.

First, download the 'Hieroglyphics.zip' file. This dataset is built from 4,210 hieroglyphs found in 10 different pictures from the book "The Pyramid of Unas" (Alexandre Piankoff, 1955). This dataset is compiled by Morris Franken, complementary to the paper titeled "Automatic Egyptian Hieroglyph Recognition by Retrieving Images as Texts" (ACM Conference on Multimedia, 2013).

The original pictures look like the following:

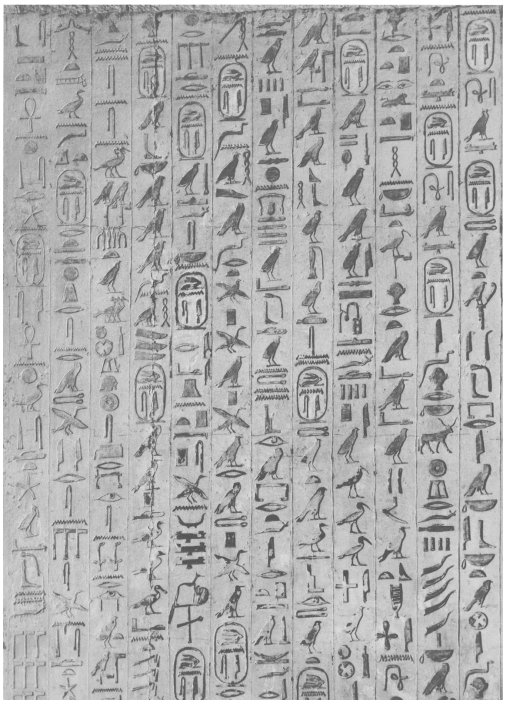

Figure 1:

The authors used a a text-detection method to extract the hieroglyphs automatically from the full pictures. After the hieroglyphs were cut out, the images were pre-processed to fit in a 50x75 pixel frame, while maintaining their width/height ratio. The remaining unfilled pixels were filled with background texture to mimic the actual background.

As you may be familiar from your CS classes, a picture can be represented as decimal number between 0 and 1 for each pixel representing how light or dark the pixel is. This enables seemless transition between what our eyes perceive as pictures and a mathematical representation that will be convenient for our analysis.

Your overall objective for this lab is to return a zip file with the hieroglyphics sorted into separate folders for each group. The coding will be fairly concise, but you will want to try several different approaches to find the best solution, decided both visually and by the statistics we discussed in class.

NOTE: We are inching towards bigger data and larger computations than you have likely worked with in classes to this point in your career. Be smart with your computation time! While coding, start small, testing your code and methods on tiny subsets of the data to ensure your methods are working correctly before scaling up to the full challenge. The last thing you want is to sit for 10 minutes while a loop runs, only to find a typo in the last line crashing the code. Another strategy I like to use is evaluating the run time of some code with the Sys.time function on my tiny subset, then scaling it to estimate the full run time (You can see an example of this in the Multidimensional MI code I posted). Finally, be smart about when running your large computations - have 1 hour of computations to run? Start it before going to dinner. Have 8 hours to run? Run it just before going to sleep for the night.

1. First start by simply looking at some of the images - how many different types of hieroglyphics do you see? This will give you some idea of a reasonable range of k values for k-means clustering.
2. Read in all the images in R and store them as a single data frame. The 'readPNG' function of the png package and the 'list.files' functions will help.

You want one row per image, one column per pixel. This will require melting the data into a different shape. You can accomplish this with 'data.frame(matrix(image, nrow=1, ncol = 50*75))', which turns the numeric array 'image' of size 50 x 75 into a one row data frame. Store both this data frame and the raw input imges, to allow us to easily write the images again when we have a solution.

3. Now start running some k-means clustering algorithms. The Cluster code I posted to Notebowl should give you everything you need for this.
4. To easily check your output, write each image to the folder corresponding to it's assigned cluster. 'dir.create(paste("Cluster",k, sep=""))' will create a folder named Cluster(whatever value k is) in your working directory. The 'writePNG' function of the png package will handle the writing for you. Loops are the way to go here.
5. Finally, write a function to predict which cluster new images belong to. (How would you pick a cluster for a new point?) The 'Predictions.zip' file contains 5 new images. For each, generate the prediction, and visually contain the image to the other images in that cluster. Does it seem like the clustering is doing a good job?

When you are satisfied with your answer, turn in your R code, a zip file with the sorted images, and a 1 page write up explaining what steps you took to arrive at your final answer. Your grade will depend on you employing sufficient effort to ensure a "good enough" solution. I want to see a thoughtful approach to constructing the final solution.

You will need to experiment with preprocessing methods such as scaling and PCA, choice of number of clusters k, and the number of random initial starts to find a solution that works best. Remember the No Free Lunch Theorem - there isn't one approach that works best for all problems, you have to play around and find what works best for your particular problem. While you can't generate a single number like MSE to tell you how your method is performing, visually comparing the clusters and examining the relevant statistics should give you some measure.