COGS 118B Final Paper

Introduction and Motivation:

Cancer has long been a notorious leading factor of mortality across the world, with colloquial phrases such as “curing cancer” to be synonymous with one of the greatest achievements of mankind. Our data analysis of the Skin Cancer MNIST: HAM10000 dataset utilizes two unsupervised machine learning algorithms - Principal Component Analysis and K-Means clustering - to gain insight into statistical factors that will make it easier to identify between seven kinds of skin lesions. Herein, we summarize our methodology, results, and findings in hopes of stimulating interest in developing new approaches to preventing and ‘curing’ skin cancer.

Related work:

According to *Cancer.org,* skin cancer is the most common type of cancer. The good news is that most skin cancers can be treated effectively if they are diagnosed early. In fact, according to skincancer.org, “More people are diagnosed with skin cancer each year in the U.S. than all other cancers combined.” There are not enough skin Cancer specialists who can accurately diagnose skin cancer. Also, millions of Americans don’t have access to quality care, which makes early diagnoses almost impossible. Machine Learning has the potential to alleviate that problem. According to a 2021 research paper from MDPI, Machine Learning can be used to “develop a system to evaluate images of the skin to diagnose skin cancer.” A Stanford Research paper from 2017 used Deep convolutional neural networks (CNNs) to classify various skin cancers, and achieved a “level of competence comparable to dermatologists.” Inspired by these related works, we wanted to take an Unsupervised Learning Approach to this problem.

Methods:

Overall data prep: After downloading the Kaggle dataset into our computers, we began by first converting the images to gray by creating a convert_to_gray function. From there, we looked at each dataset provided and determined which ones were of value to us. We used pandas to create a dataframe out of the 'HAM10000_metadata.csv' which shows the identity of each lesion image. We then created another dataframe out of ‘'hmnist_28_28_L.csv' which has the pixel values of each image plus a numerical label that identifies the type of lesion in the image. We removed the label column from the dataframe so that we can properly run our algorithms on the dataset.

K-Means Clustering: To apply Kmeans to the given dataset, we needed to create the functions runKMeans. The function would begin by setting X as the dataframe and N, D as np.shape(X)[0] and np.shape(X)[1], respectively. We initialized cluster centers as Kmus, which are random points in the data. We also needed to create functions that would act as components of the runKMeans function. We began by creating a calcSqDistances function which will create a square distance matrix given the dataset and an initial Kmus. We then created the function determineRnk which would assign each datapoint's closest centroid given the square distance matrix from the previous function. We then created a recalcMus function which would recalculate the mu value given the dataset and the assignment from the previous function. We would iterate through these three functions within the runKMeans function until the centroid converged, at which point we would break and return the updated Kmus value.

Principal Component Analysis: The dataset that we used has 10015 28 by 28 images. This would result in each image having 784 dimensions. Therefore, we decided to perform a Principal Component Analysis (PCA) on the dataset in order to decrease the number of dimensions. By using PCA, we will be able to determine which features contribute the least, and then drop them. We first calculated the mean of the data points, created a matrix A consisting of all mean subtracted data points, then used matrix A to construct a covariance matrix. Next, we used the covariance matrix to determine the corresponding eigenvalues and eigenvectors. In order to compute these eigenvalues and eigenvectors, we used the linalg.eig function from NumPy. The eigenvectors represent the orthogonal axes (or principal components) of the direction of the new subspaces while the eigenvalues represent the magnitude (variance) of the eigenvectors. This is important since the goal of PCA is to find a subspace (which in this case are the eigenvectors) such that the orthogonal projection of data points on it maximizes the variance of the projected points. Therefore, we would need to sort the eigenvalues and their corresponding eigenvectors in order of largest to the smallest. This is because for PCA, we are only interested in the top k principal components. For our project we have decided to use the top 4, 8, 14, and 20 principal components for this step. In order to transform the dataset using the k amount of dimensions, we then used matrix multiplication to multiply matrix A by the matrix of sorted top k eigenvectors to get the transformed data. The aim of this step is to use the selected principal components (k = 4, 8, 14, and 20) formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components. The results of these reoriented data will be discussed in the next section.

Results:

PCA: From creating a scree plot of the principal components, we were able to visualize an elbow curve that demonstrated how much variance each principal component explained in the orthogonal projection of data points. We were able to determine that principal components 1 through 14 captured the most significant amounts of variance, and after the 14th principal component, the amount of variance explained tapers off to an inconsequential amount. We confirmed our interpretation of the scree plot by projecting multiple different images (the 70th and 101st image) of skin cancer onto the subspace spanned by the principal components. When we only used 4 or 8 principal components, the reconstructed images did not look like the original image; rather, they appeared to be distorted images of the mean image calculated across the entire data set of images. On the other hand, when we included more principal components past our decided significant point of 14 principal components (we used 14 and 20 principal components in two separate tests), we were able to reconstruct images that still captured characteristic features of the original images that allowed the reconstructed images to look similar to the original images. Thus, we can conclude that using PCA, including interpreting the scree plot, worked well in order to create a subspace for the images that reduced dimensions. However, during our process of conducting PCA, we had difficulties successfully compiling our code in order to perform necessary calculations. At first, we were concerned there was an issue with our code that was causing a runtime error, and then we identified that the issue was calculating the eigenvalues; it was taking a long time to run because of how many images were in our dataset. We were worried that Google Collab would not be able to successfully complete these calculations, but after 11 minutes the calculations were completed and we were able to continue our PCA process without any other issues.

K-Means Clustering: We originally attempted to use the original dataset resolution, of 450 by 600 in color, however the resulting dataset of dimensions 5000 images by 450 by 600 by 3 was too large for our computational resources to handle. Because of this we attempted to convert the RGB image into a Grayscale image, reducing the dimensionality of each image from 3 to 2 dimensions, and the dataset from 4 to 3 dimensions. As this still required too much memory, we downsampled the dataset from 450 by 600 to 113 by 150 using OpenCV pyramid downscaling. After all the hassles of retrieving datasets from kaggle, converting between ndarrays and pandas dataframes, and other issues revolving around invalid dimensions and code errors, we resorted to using a 28 by 28 image size of 10015 images which were already complied

in the given CSV files. After running KMeans on our dataset of dimensions 10015 by 28 by 28, it iterated 38 times and produced 7 different cluster plots.

Due to the large size of each image and number of images, the preprocessing of images and compiling into a numpy array took around 10 minutes to complete. The Kmeans clustering into 7 clusters shows mainly the sizes and the centers of each of the lesions on skin.

Discussion:

PCA: We learned that using between 14 to 20 principal components, gave us the best recreation to the original image. One thing we could have done better is have a more efficient algorithm to calculate the eigenvalues of A^TA. This took 11 minutes because the result was a 10,015 x 10,015 matrix since the dataset was so large. Also, PCA assumes all the eigenvalues are independent of each other. It is likely that a black pixel is next to a black pixel, so it is dependent. This is a disadvantage of using PCA in our situation. One extension to improve our project is to have a smaller training dataset by randomly choosing some of thousands of images. This is because the mean skin cancer image we created looked like a circle in the middle with it being darkest in the center and lightest on the outside. If we used a subset of the total 10,015 images, we could have a more interesting mean skin cancer image, which potentially could lead to better results. Another extension we could have done is worked with images with color. We wanted to reduce the complexity of the image, so we converted the image to grayscale. Using colored images could potentially lead to more interesting results as the color of skin cancer can be quite visibly stark in comparison to the rest of the skin. Furthermore, we can also look into more diverse skin cancer datasets. Most of the images in our dataset were of caucasian individuals, which may mean our results may not be applicable to people of color.

K-Means: Interestingly enough, we found that the clusters seemed to be formed on the basis of size rather than the shape of our lesions. As our data had been downsampled from 600 x 450 RGB images to 28 x 28 grayscale images, there is obviously a very large loss of data through that conversion. With colors only being in greyscale, there are also issues where brightness of the image may also act as a confounding factor to the analysis and generation of our clusters; color variation and pigmentation may be mistaken for brightness. Looking at the results of our K-Means clusters, we had previously noted that our clusters appeared to have been inverted. Rather than darker values on the inside of our clusters, it was the opposite. As a result of performing K-means on grayscale images where darker values may have been the majority of

pixels in our images, we suspect the clustering algorithm to prioritize lighter values to form the clusters. This is something worth noting for future image-related data analysis involving K-means algorithms, as it could lead to very unexpected results (like ours!). Additionally, higher resolution scaling would allow for more accurate results; with more computational resources we could generate more accurate findings and allow for the incorporation of more expansive datasets. Given more time, we could have re-evaluated our overall approach for our K-Means algorithm and processed our images differently to avoid inaccurate clusters.

Contributions:

Prothit Halder (A15851067): Worked on the PCA code, did the Discussion PCA section and Related Works section, and facilitated meetings.

Natalie Kwong (A15944830): Worked on the PCA code, did the Results PCA section, video recording/processing

Jota Yamaguchi(A15581388): Worked on K-means code, did the Methods K-means section

Helen Zhao (A15919169): Worked on the PCA code, did the Methods PCA section

Sandesh Shrestha (A16654824): Worked on preprocessing data, K-Means code, K-Means Results & Discussion

Nolan (A16653842): Worked on K-Means code, did Introduction and Discussion for K-Means

Code: https://github.com/nataliekwong925/cogs118b-final-project

Video Link: https://youtu.be/-WphhmtxO7E

References:

- Skin Cancer Image Gallery | Photos of Skin Cancer
- https://www.skincancer.org/skin-cancer-information/skin-cancer-facts/
- **MDPI Paper:** Das, K.; Cockerell, C.J.; Patil, A.; Pietkiewicz, P.; Giulini, M.; Grabbe, S.; Goldust, M. Machine Learning and Its Application in Skin Cancer. Int. J. Environ. Res. Public Health 2021, 18, 13409. https://doi.org/10.3390/ijerph182413409
- **Stanford Paper:** Esteva, A., Kuprel, B., Novoa, R. et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056