COGS 118B Final Project Report

Deze Lyu (A15572594) Muchan Li (A15887258)
Anni Li (A16157145) Chenggong Li (A16008692)

Introduction and Motivation

Facial emotion recognition is an interesting and useful application of machine learning. In our COGS 118B class, we have practiced how to generate the mean face and how to use PCA to perform dimensionality reduction over a set of face images. In this project, we hope to extend what we have learned in class, using PCA with a K-means clustering approach to train our model to perform the facial emotion classification, as opposed to performing a brute-force K-means clustering on all dimensions of the image pixels, where the latter would be extremely computationally demanding and inefficient. By working on this project, we hope that we can gain some insight on whether by applying PCA with K-means, our program is able to detect and assign similar high-dimensional facial expressions data (as in its emotion) together, and if so, would it capture the correct emotion patterns better than just randomly assigning images to an emotion category. This project also provides another valuable opportunity for us to connect and practice what we learned in the later half of the class, strengthening our coding skills and getting a better understanding of unsupervised learning.

Related work

In practice, a data set would inevitably contain many irrelevant features, high dimensions, and lossy information, which can disturb and affect the performance of K-means clustering, making the clusters blurred and not accurate for predictions. One method to solve this problem is to operate the data by PCA before feeding them to K-means clustering. PCA can be used to project data to a lower dimensional subspace and K-means is then applied in the subspace (Zha et al., 2001). By performing PCA, we can reduce the dimensionality of the image, adding different weights to different components of our data so that important features are extracted to be focused on. There are many existing works showing that PCA is useful for unsupervised dimension reduction and effective in improving the predictive accuracy of K-means clustering. In *K-means Clustering via Principal Component Analysis* (2004) of Chris Ding et.al., the authors have shown that as dimensions are reduced by PCA, the cluster accuracy improves from 75% at 1000-dim to 91% at 5-dim (Ding et al., 2004). Another work done by Turgay Celik in 2009 also shows that PCA is an effective way to improve K-means clustering results. They used PCA to generate eigenvector spaces on an unsupervised binary-classification model, aiming at detecting changes in geographical maps. The result shows that the proposed algorithm performs quite well on combating both the zero-mean Gaussian noise and the speckle noise (T. Celik, 2009), meaning that the performance of K-means clustering is significantly improved.

Methods

First, we find, download, and import the dataset from Kaggle. It is important to note that since we are approaching this project through unsupervised means, we preprocess the entire raw dataset instead of doing any train-test splits since making predictions is of lesser concern in unsupervised models. We store the face images and the corresponding emotions into two arrays. At first, we were using the first one thousand pieces of data, but then we realized that the emotions are not evenly distributed, so we write a filtering function that selects the same number of images of different emotions, which is important to produce a balanced result for our clustering algorithm.

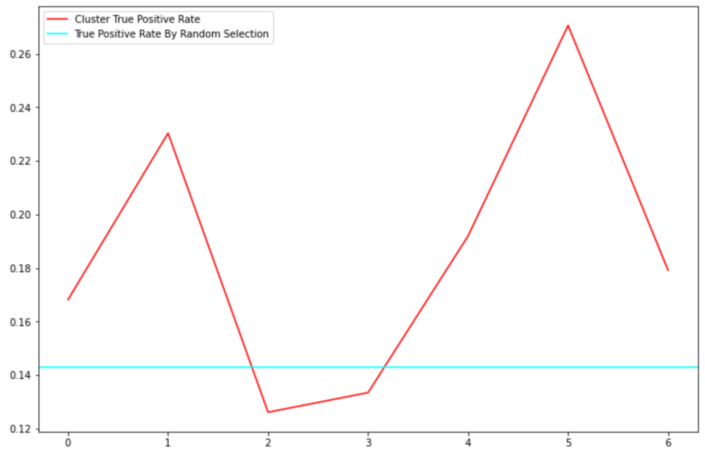
Second, we perform the PCA transformation on the dataset. We first obtain the mean image using the function np.mean(), then calculate the covariance matrix. Modifying the eigsort function provided in Homework 5, we calculate the eigenvalues and eigenvectors while having the eigenvalues arranged in descending order and the eigenvectors arranged correspondingly with respect to its associated eigenvalue. Our objective here was to select the first N principal components (i.e. the eigenvectors we obtained) such that the total sum of variance explained by these N principal components would explain at least 95% of the total variance–this 95% threshold is set by convention. We graph a scree plot to help us visualize and gain some intuition about the amount of variance explained by each principal component up till the first 50 principal components. Through some calculation, we decide to take the first 223 principal components for which the sum of variance explained exceeds 95% of total variance, thus we reduce the dimension of our dataset from 2304 to 223 dimensions.

Then, we run the K-Means algorithm. We run K-means iteratively to assign images to clusters based on their updated squared distance from the centroids. We first randomly assign the centroids as a few training set images. Then we iteratively perform the following steps: we calculate the minimum sum of the squared distance of each figure from the cluster centers and generate the square distance matrix. We then obtain the rank of each image by assigning it to the cluster center having the least squared distance apart. Then we can obtain new cluster centroids. Until the two iterations differ insignificantly, we stop the process and record our final result. We choose seven clusters to implement the K-Means algorithm because there are seven main emotions in total–so we are hoping that the K-means algorithm would uncover the intrinsic variability in seven emotions and cluster images that belong to each emotion as a group without explicit labels (even though we have them).

At last, we reconstruct the images described by the seven cluster centers to see if they are representative of any of the seven emotions. Besides this, we check our cluster assignments through two approaches: on one hand, we check the proportion of each emotions contained by each cluster so that we could gain intuition as for which cluster captures roughly which emotion; on the other hand, we calculate the true positive rate of emotion detections based on the main cluster emotion label we just uncover. We also visualize some images in the two of our best performing clusters. All results will be discussed and displayed in the following section.

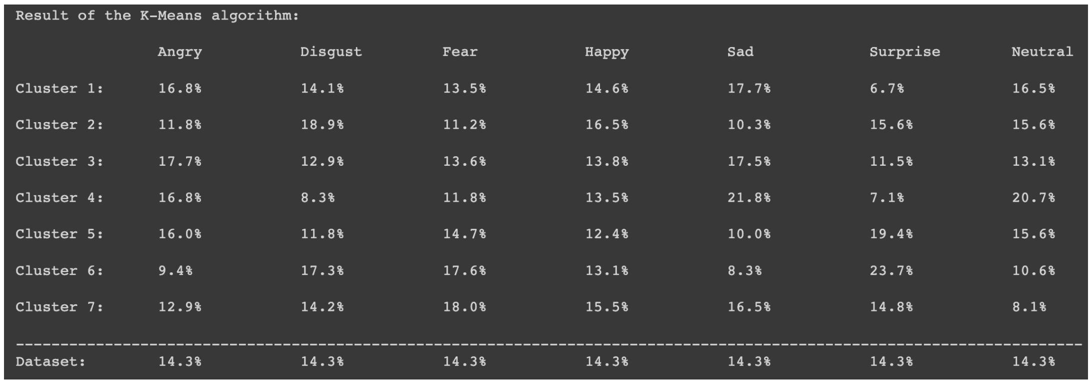
Results

Cluster True Positive Rates:



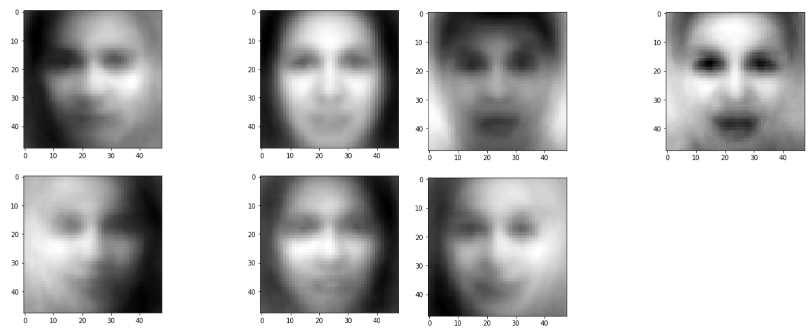
For the graph above, the x-axis lies our emotions numerically, the values are discrete where 0 is for angry, 1 is for disgust, 2 is for fear, 3 is for happy, 4 is for sad, 5 is for surprise, and 6 is for neutral. The y-axis represents proportions in decimal. It indicates that surprise emotion has the highest true-positive assignments based on the images assigned to the surprise cluster from K-means, at around 27 to 28 percent; disgust emotion follows, with true-positive assignment at around 24%. Overall, we have 5 emotions–angry, disgust, sad, surprise, and neutral–achieving better true-positive assignment proportions than assigning image to emotion at random, which would be 14.3%, as highlighted by the cyan line. However, this does not mean that our clusters yielded by K-means algorithm uncovered extremely useful information since on one hand, we couldn't really devise a way to see if our performance is statistically significant compared to the null, random performance. On the other, the true-positive results are simply not predominant, as we were expecting around 80% true positive results before implementing and running the algorithm.

Percent of Each Emotion in K-Means Clusters:



```
Result of the K-Means algorithm:

                Angry           Disgust         Fear            Happy           Sad             Surprise        Neutral

Cluster 1:      16.8%           14.1%           13.5%           14.6%           17.7%           6.7%            16.5%

Cluster 2:      11.8%           18.9%           11.2%           16.5%           10.3%           15.6%           15.6%

Cluster 3:      17.7%           12.9%           13.6%           13.8%           17.5%           11.5%           13.1%

Cluster 4:      16.8%           8.3%            11.8%           13.5%           21.8%           7.1%            20.7%

Cluster 5:      16.0%           11.8%           14.7%           12.4%           10.0%           19.4%           15.6%

Cluster 6:      9.4%            17.3%           17.6%           13.1%           8.3%            23.7%           10.6%

Cluster 7:      12.9%           14.2%           18.0%           15.5%           16.5%           14.8%           8.1%

-------------------------------------------------------------------------------------------------------------------------
Dataset:        14.3%           14.3%           14.3%           14.3%           14.3%           14.3%           14.3%
```

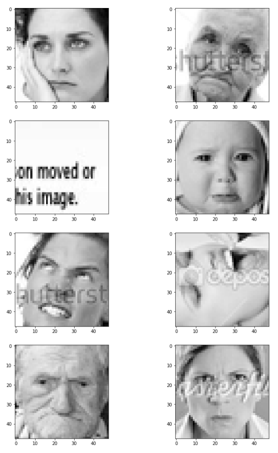
Reconstructed Image Using Optimal Centroids From K-Means



After obtaining the seven centroids from running the K-Means algorithm, we calculate the proportion of each emotion in each of the 7 clusters, as shown in the first table. Based on its results, we first decide cluster 6 to be the surprise cluster since 23.7% of the images in it are of surprise emotion. The remaining clusters have almost even split between emotions, but with reference to the images we reconstructed using only the 223 principal directions, we concluded that cluster 1 would be for neutral expressions, cluster 2 for disgust, cluster 3 for angry, cluster 4 for sad, cluster 5 for fear, cluster 6 for surprise (as expected), and cluster 7 for happy.

We also visualize and contrast the images contained in our surprise and disgust clusters given that they are our best-performing clusters. Due to the volume of images generated, we would not include them in this report; however, you can click on the link to our Google colab notebook, scroll to the end, and you will see the set of images.

Discussion



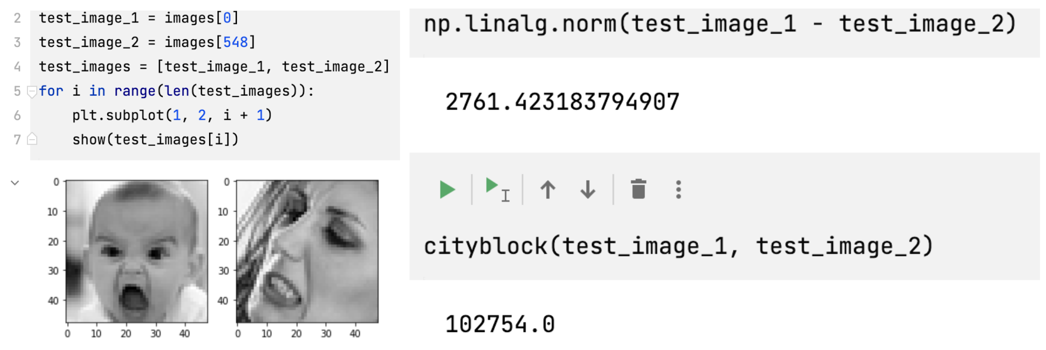
One problem we encounter in the project has to do with the dataset itself. There is too much noise among the images. As shown on the right, for example, some of the images here have watermarks; one is improperly rotated; one contains only text; and some seem to be incorrectly labeled (these images are supposedly angry, but it seems to us that image 1 and 4 as examples look more sad than angry). Therefore, if we had more time or had we started even earlier, we would devote more labor toward screening the images, removing unfit labels, and adjusting rotations as necessary. We believe this would set us in a cleaner direction.

There are also several weaknesses in our approach: first, as we learned afterward, due to the curse of high dimensionality, K-Means algorithm can hardly calculate significant deviation in Euclidean distance between images, thus making it difficult to distinguish one emotion from another. In order to preserve minimal loss of information, we selected 223 principal directions, reducing our data to occupy a 223-dimensional space. While 223-D may seem like a huge improvement from the original 2304-D, it is still high-dimensional for running K-Means. Hence it should be of little surprise that K-Means would not capture the underlying pattern as we expect it would.

Secondly, It is also hard to create a standard for getting the best number of clusters for K-Means. Most ideally, since there are seven kinds of emotions, the model should be trained to come up with seven distinct clusters, one representing each emotion. In this case, each cluster would have a rather high percentage for a certain emotion. Thus why we chose seven clusters in total. In reality though, not all emotions can be clearly classified into one single cluster. One such example would be image 1 shown on the set of images on the previous page–it could be clustered into either sad or angry. Thus, exactly seven clusters might not be the most optimal number of clusters. From adjusting the number of k-cluster parameters, we learned that sometimes there is a tradeoff between getting a precise number of clusters and getting better specificity in each cluster. We choose seven clusters, which best fits our intuition in this project. If we have more time, we might obtain a better result by doing more research on choosing the best number of clusters in the K-Means algorithm.

We then propose two perspectives of improvements we can later revisit upon. In response to our first weakness, we would thoroughly clean our image dataset before deploying PCA. Eliminating unnecessary noise in our data would significantly bring the number of distracting factors down, thus we would arrive at a lower number of principal components to achieve 95% of total variance. We would also research on the topic of tradeoff between choosing less principal components and capturing less variance to obtain an optimal tradeoff point. We would also try implementing a function that calculates the Manhattan distance between vectors. As a side lesson from our research, we learned that calculating the Manhattan distance between two high-dimensional observations would be more effective and discernable than calculating the ordinary Euclidean distances. This calculation method should be able to make the K-Means algorithm more "sensitive" to deviations in distance, making the situation better. In response to our second weakness, we would need to optimize the number of clusters first using either elbow method or average silhouette method. This way we would gain some quantitative intuition on which K is better and how much difference it would make between choosing the more intuitive K (such as our case where K is set to 7 merely because we have 7 emotion categories) and the optimal K.

We believe that we can detect the underlying structure of diverse emotions more accurately by calculating the Manhattan distance with a demonstration below:



```
test_image_1 = images[0]
test_image_2 = images[548]
test_images = [test_image_1, test_image_2]
for i in range(len(test_images)):
    plt.subplot(1, 2, i + 1)
    show(test_images[i])
```

```
np.linalg.norm(test_image_1 - test_image_2)
```

```
2761.423183794907
```

```
cityblock(test_image_1, test_image_2)
```

```
102754.0
```

The two images are each chosen from the angry group and the disgust group, so it is safe to assume that their underlying pixels and information differ significantly. However, as shown by the first calculation on the top right, which is taking the Euclidean distance of the two, their distance is at around 2761 across 2304 dimensions. One would almost think that the two images are nearly the same since the average variation is barely above 1 unit per dimension. In contrast, the Manhattan distance calculated on the bottom right yields distance value greater than 100,000, indicating significant dissimilarity.

Contributions

Deze Lyu: I created the Google Colab document and developed a foundation for our final project, which allows us to import and filter the dataset, perform PCA transformation on the selected data, run the K-Means algorithm, and visualize the result of the clusters by printing out relevant images and statistics. I encountered several problems when coding the EM algorithm, and that's where I stopped. I also participated in writing this document and recording my part in the video.

Anni Li: Background research; find the raw dataset; run & test the code and debug; edit some details of the code in general; write the introduction section, the literature review section, and part of the discussion section; final review of the report and code.

Chenggong Li: Running and testing the code. Trying different numbers of clusters for the K-Means algorithm and choosing the optimal parameters. Writing part of the methods and discussion section.

Muchan Li: incorporated K-Means and PCA components together (before Deze just ran them individually to see what the dataset would output); computed true positive rates in each K-Means cluster outputs; produced most of the visualizations in the notebook; wrote results and the majority of discussion section of this report; edited the group video since we recorded in individual pieces; created the presentation slides (even though in our video we presented primarily based on our Jupyter notebook).

Code

Our code is in a Google Colab document. To reproduce the result, please follow the instructions in the beginning of the document to download and import the dataset.

References

"Challenges in Representation Learning: A report on three machine learning contests." I Goodfellow, D Erhan, PL Carrier, A Courville, M Mirza, B Hamner, W Cukierski, Y Tang, DH Lee, Y Zhou, C Ramaiah, F Feng, R Li, X Wang, D Athanasakis, J Shawe-Taylor, M Milakov, J Park, R Ionescu, M Popescu, C Grozea, J Bergstra, J Xie, L Romaszko, B Xu, Z Chuang, and Y. Bengio. arXiv 2013.

Zha, H., Ding, C., Gu, M., He, X., & Simon, H. (2001). Spectral relaxation for K-means clustering. Advances in Neural Information Processing Systems 14 (NIPS'01), 1057–1064

Ding, C., He, X. (2004). K-means clustering via principal component analysis. Proceedings of the twenty-first international conference on Machine learning (ICML '04). Association for Computing Machinery, New York, NY, USA, 29. DOI:https://doi.org/10.1145/1015330.1015408

T. Celik, "Unsupervised Change Detection in Satellite Images Using Principal Component Analysis and K-Means Clustering," in IEEE Geoscience and Remote Sensing Letters, vol. 6, no. 4, pp. 772-776, Oct. 2009, doi: 10.1109/LGRS.2009.2025059.
Bishop, C.M. Pattern Recognition and Machine Learning. Springer, 2006. ISBN-13: 978-0387-31073-2