Facial Recognition with Linear Systems

MATH 365: Technical Report 4

David Trinh, Jefferson Cunin, Roger Carranza

Spring 2025, Macalester College

Abstract

This technical report contains an overview of Singular Value Decomposition (SVD), projection matrices, and an application of this in the context of facial recognition using multiple images associated to human emotions. This is done through a conceptual introduction of SVD, its manipulation; and lastly, its role in generating insights for this case. The objective of this application is to understand which emotion from the available data we are the most accurate at identifying using linear algebra methods, training and testing splits.

Overview of Singular Value Decomposition

Singular Value Decomposition allows for the factorization of any matrix A of dimensions $m \times n$ so it is expressed as a function of two orthonormal bases respective to A's domain and range. It relies on the manipulation of $A^T A$ and the general representation for full decomposition is shown as follows;

$$A = U \ \Sigma \ V^T$$

Where V is an orthonormal matrix of dimensions $n \times n$ whose column vectors are orthonormal eigenvectors of the matrix $A^T A$. Σ is a diagonal matrix extended with zeros depending on the original matrix; its dimensions are $m \times n$ and the diagonal entries are the singular values of A, which are the square root of the eigenvalues of $A^T A$. Lastly, U is an orthonormal matrix of dimensions $n \times n$ whose column vectors are defined by the following expression;

$$u_i = \frac{1}{\sigma_i} A v_i$$

This is relevant not only because of the multiple applications that we can find based on the properties of orthonormal matrices such as U and V, but also because of the information embedded into each of them. In the case of U for example, it is constructed by applying A to $A^T A$ eigenvectors, which being scaled by their corresponding singular values results in a set of orthonormal vectors. Therefore, those vectors represent the information of our original matrix A aligned with the directions A sends $A^T A$ eigenvectors into. Given that the singular values are sorted up in descending order, their magnitude is associated to the relevance of their corresponding vectors in U and V– which are analogous to each other as described before– in capturing the information of A. Bear this idea in mind because it is particularly relevant for the application contained in this report.

Considerations of SVD

It is worth mentioning that in order to successfully decompose a matrix A into its SVD form, we must consider its properties and structure because the result will depend on it. Firstly, consider the rank r of the matrix A. Given that $A^T A$ has as many non-zero eigenvalues as the rank r of A, we may not be able to

obtain a full basis of U by simply projecting the column vectors v_i onto A and scale them by their respective singular values. A solution for this is to extend it from the available column vectors up until u_r using the Gram Schmidt algorithm. In Summary, all there to do is to project the available u_i vectors onto one another and subtract from the original vector we used for the projection, which will result in an additional orthogonal vector; iterating through this we can extend the dimensionality of U so it has dimensionality $m \times m$.

Another consideration is that in the case of full decomposition, the matrix Σ can be thought of as a diagonal matrix $n \times n$ of singular values σ_i, to which we bind columns or rows of 0's depending on whether the original matrix A is a tall matrix $m \times n$ such that $m \geq n$, or a wide matrix $m \times n$ such that $m \leq n$. The point of this is so that Σ matches the dimensionality of its adjacent matrices U and V.

Data

The data for this report was obtained from Kaggle. It contains 7 folders, where each represents an emotion from the list; Anger, Disgust, Fear, Happiness, Sadness, Surprise, Contempt. Each of the files contains multiple images in png format of human faces intending to show the facial features associated with each of the respective emotions. For example; a random image from the Happiness file will contain a smile. The pictures are in gray scale so there is no need to transform the color scheme of the original data.

Methodology

Given that our data set is in the format of multiple png files, our first objective is to encapsulate all images per emotion into a matrix; for example, "anger_matrix". This can be done by flattening each of the individual images into a vector; recall that each image is a matrix with each entry representing color intensity of an individual pixel, so the flattened vector will be its entries per row in consecutive order. Next step is to use each of those vectors as columns of its respective emotion matrix. It is worth noting that we count with different amounts of files per emotion, for which our different emotion matrices differ in dimension; this also implies that given our objective of splitting the data into training and testing folds, the matrices containing more information– Or columns representing a file each– might return lower residuals.

As a result of the description above, each 'emotion matrix' has as its column space the information captured by each individual image it contains. Of course, this is before the split into training and testing folds that will be done for each emotion matrix; the split is 90% of information in training fold and 10% for testing.

Once the splitting has been done, we will use the training fold to build what can be considered as a 'model' for each emotion matrix. Lastly, we will use the images from the testing fold on the 'model' of each respective emotion to identify which one results in the lowest average residual.

Below it can be observed how the dimensions vary for each emotion, which is a consequence of data availability.

```
dim(anger_images)
```

```
## [1] 2304  135
```

```
dim(contempt_images)
```

```
## [1] 2304   54
```

```
dim(disgust_images)
```

```
## [1] 2304  177
```

```
dim(fear_images)
```

```
## [1] 2304   75
```

```
dim(happy_images)
```

```
## [1] 2304  207
```

```
dim(sadness_images)
```

```
## [1] 2304   84
```

```
dim(surprise_images)
```

```
## [1] 2304  249
```

The Model - Why this works

What is meant with 'model' refers to the manipulation of each emotion matrix respective training fold A_i using SVD. Recall what the structure of each emotion matrix is like, being composed of several images in the form of collapsed vectors as their column space. Then, for the matrix A_i its SVD from presents as follows;

$$A_i = U_i \ \Sigma_i \ V_i^T$$

Since U_i contains the eigenvectors of $A^T A$ projected onto A_i scaled by their respective singular values. Then the matrix U_j is the best j dimensional basis that captures the most information from the original data. So assume we wanted to reconstruct an image from outside the training fold A_i using its information; this can be done by constructing the projection matrix of U_j using the general projection matrix formula;

$$P = U(U^T U)^{-1} U^T$$

A projection matrix is defined such that for a vector b that is not in the span of P, Pb is the closest vector to b within the matrix P span. But since U is orthonormal and satisfies $U^T U = I$, the expression above reduces into the following;

$$P = U_j \ U_j^T$$

So for an image in vector format v, Pv is the closest our training fold from the emotion matrix builds it back. Lastly, We can measure this by checking the residual using norms, so for a vector v, the residual is defined as

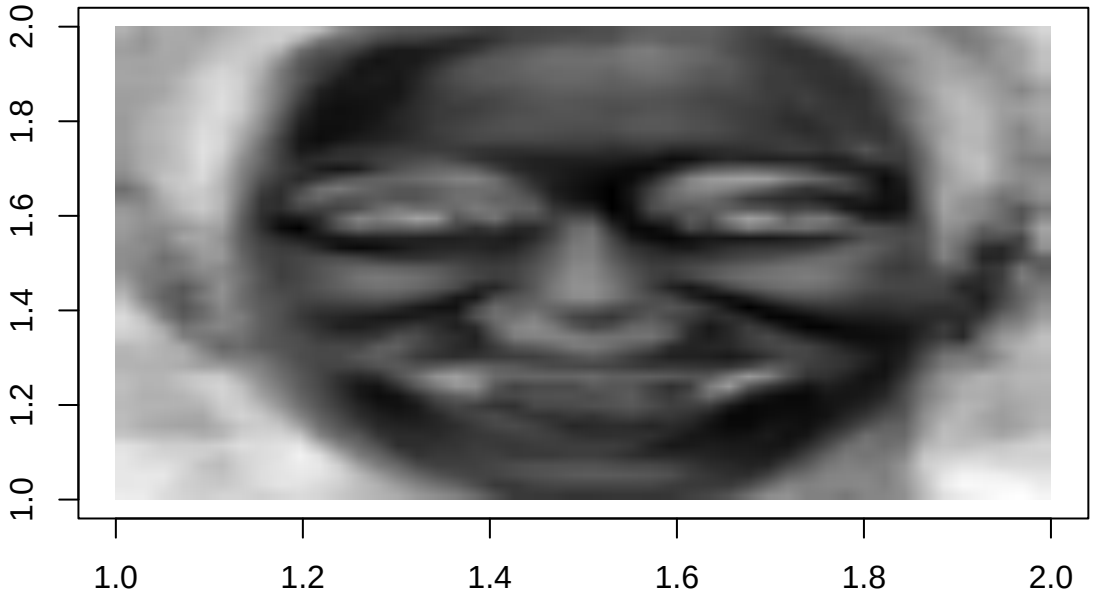
$$r = v - Pv = (I - P)v$$

But before applying this on each respective training fold, once the split has been done it is crucial to center the data for each set of images. This means subtracting the mean image from each image in the set. The purpose of this is that SVD does not simply redirect us to the mean image of our data; but rather, to aim at its underlying structure. Analogous to this particular case; we do not want SVD to simply consider as the standard of facial features to have a nose in the center of the image, or lips with a similar width; we want it to identify less obvious patterns, such as eyebrows in a particular angle, or eyes with a slight bent.

Lastly and summarizing; for each emotion matrix A_i we will split into training and testing folds, use SVD on training fold to obtain matrix U_i, get the projection matrix of U_i, and then calculate the average residual norm of all image vectors v_i from the testing fold.

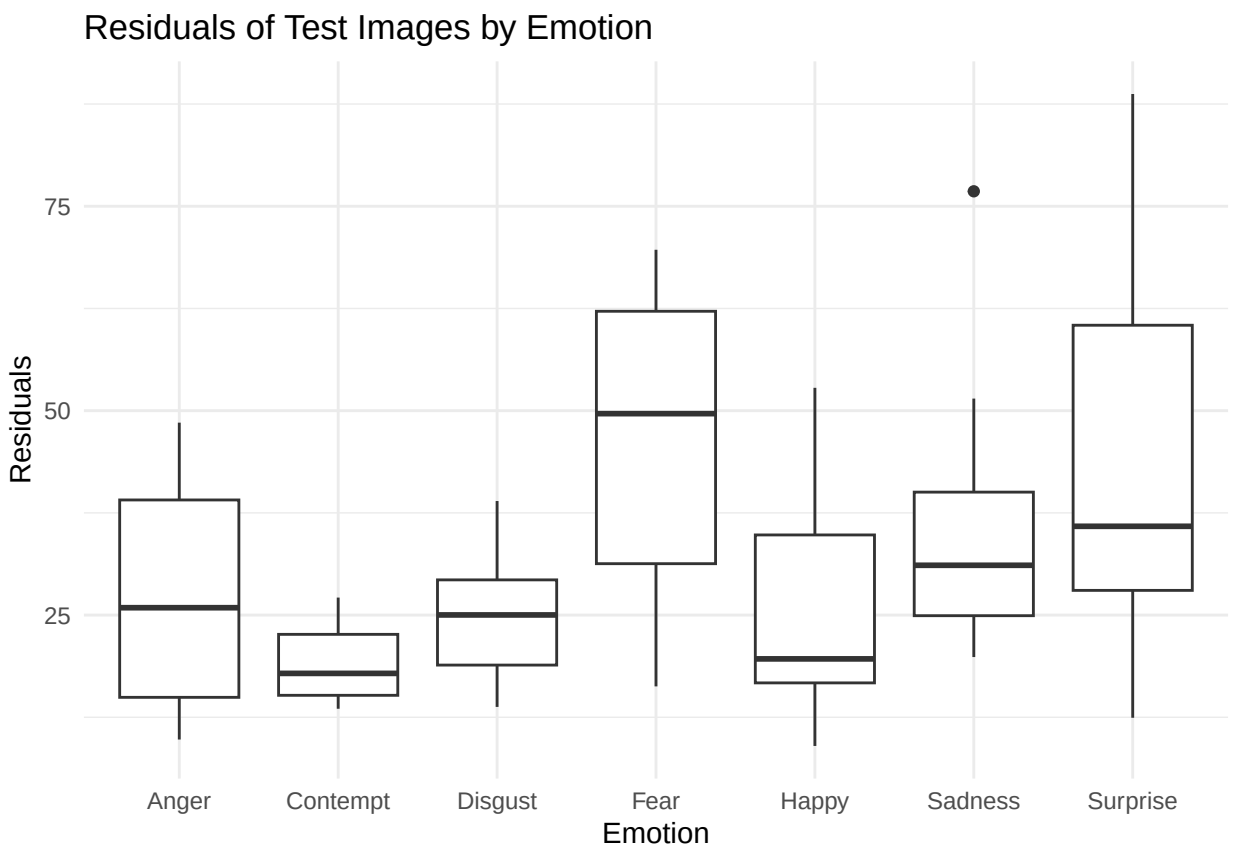
Observe an example of an image representing the U_1 of our happiness training fold;

```
imgprinter(happy_U[, 1])
```



Conclusion

As observed by the box plot below, the emotions associated to the lowest residuals are Contempt, Disgust and Anger. While we lack the biological-psychological literature to attempt to justify our results in terms of what is particular to each emotion, we can say that within the constrains of the available data, humans show a varying degree of variability in representing each of the different emotions used for this project. We can also conclude on the fact that SVD is effective at capturing trends within an information set; in this case, the physical appearance of the multiple pieces of a human face that can vary and hence, be reflected in pixel format. Whether it is a particular set of muscles being pulled around the eyebrows, or a slight bend in the smile; if it can be put into matrix format, SVD can identify its underlying structure.



External References

1. Images source, Kaggle, https://www.kaggle.com/datasets/shuvoalok/ck-dataset
2. Code snippet for loading png into floats, magick R package https://cran.r-project.org/web/packages/magick/vignettes/intro.html, and ChatGPT.

Appendix

```
imPlot = function(img,...) {
    plot(1:2, type='n',xlab=" ",ylab= " ",...)
    rasterImage(img, 1.0, 1.0, 2.0, 2.0)
}
imgprinter = function(v){
    imPlot(Reshape(colorshift(v),48,48))
}
```

```
image_to_matrix <- function(file) {
    img <- image_read(file)
    img_gray <- image_convert(img)
    img_array <- as.numeric(image_data(img_gray))
    matrix(img_array, ncol = 1)
}

# Remove redundant image
remove_redundant_images <- function(images) {
```

```
    images[, seq(1, ncol(images), by = 3)]
}

# Center the images
center_image <- function(image) {
    p = ncol(image)
    meanImage = rowSums(image)/p
    meanImageMatrix = meanImage %o% rep(1, p)
    centeredImage = image - meanImageMatrix
    centeredImage
}

preprocess_images <- function(images) {
    #images <- remove_redundant_images(images)
    images <- center_image(images)
    images
}
```

```
#Loading the images, ChatGPT helped with this
anger_files <- list.files("archive/anger", pattern = "\\.png$", full.names = TRUE)
contempt_files <- list.files("archive/contempt", pattern = "\\.png$", full.names = TRUE)
disgust_files <- list.files("archive/disgust", pattern = "\\.png$", full.names = TRUE)
fear_files <- list.files("archive/fear", pattern = "\\.png$", full.names = TRUE)
happy_files <- list.files("archive/happy", pattern = "\\.png$", full.names = TRUE)
sadness_files <- list.files("archive/sadness", pattern = "\\.png$", full.names = TRUE)
surprise_files <- list.files("archive/surprise", pattern = "\\.png$", full.names = TRUE)

# Preprocess the images
anger_images <- preprocess_images(do.call(cbind, lapply(anger_files, image_to_matrix)))
contempt_images <- preprocess_images(do.call(cbind, lapply(contempt_files, image_to_matrix)))
disgust_images <- preprocess_images(do.call(cbind, lapply(disgust_files, image_to_matrix)))
fear_images <- preprocess_images(do.call(cbind, lapply(fear_files, image_to_matrix)))
happy_images <- preprocess_images(do.call(cbind, lapply(happy_files, image_to_matrix)))
sadness_images <- preprocess_images(do.call(cbind, lapply(sadness_files, image_to_matrix)))
surprise_images <- preprocess_images(do.call(cbind, lapply(surprise_files, image_to_matrix)))
```

```
set.seed(365)

# Splitting the data into training and testing sets
test_anger_indices <- sample(seq_len(ncol(anger_images)), size = 0.1 * ncol(anger_images))
test_anger_images <- anger_images[, test_anger_indices]
train_anger_images <- anger_images[, -test_anger_indices]

test_contempt_indices <- sample(seq_len(ncol(contempt_images)), size = 0.1 * ncol(contempt_images))
test_contempt_images <- contempt_images[, test_contempt_indices]
train_contempt_images <- contempt_images[, -test_contempt_indices]

test_disgust_indices <- sample(seq_len(ncol(disgust_images)), size = 0.1 * ncol(disgust_images))
test_disgust_images <- disgust_images[, test_disgust_indices]
train_disgust_images <- disgust_images[, -test_disgust_indices]

test_fear_indices <- sample(seq_len(ncol(fear_images)), size = 0.1 * ncol(fear_images))
test_fear_images <- fear_images[, test_fear_indices]
train_fear_images <- fear_images[, -test_fear_indices]
```

```
test_happy_indices <- sample(seq_len(ncol(happy_images)), size = 0.1 * ncol(happy_images))
test_happy_images <- happy_images[, test_happy_indices]
train_happy_images <- happy_images[, -test_happy_indices]

test_sadness_indices <- sample(seq_len(ncol(sadness_images)), size = 0.1 * ncol(sadness_images))
test_sadness_images <- sadness_images[, test_sadness_indices]
train_sadness_images <- sadness_images[, -test_sadness_indices]

test_surprise_indices <- sample(seq_len(ncol(surprise_images)), size = 0.1 * ncol(surprise_images))
test_surprise_images <- surprise_images[, test_surprise_indices]
train_surprise_images <- surprise_images[, -test_surprise_indices]
```

```
# SVD and u matrix per emotion
anger_svd <- svd(train_anger_images)
contempt_svd <- svd(train_contempt_images)
disgust_svd <- svd(train_disgust_images)
fear_svd <- svd(train_fear_images)
happy_svd <- svd(train_happy_images)
sadness_svd <- svd(train_sadness_images)
surprise_svd <- svd(train_surprise_images)

anger_U <- anger_svd$u
contempt_U <- contempt_svd$u
disgust_U <- disgust_svd$u
fear_U <- fear_svd$u
happy_U <- happy_svd$u
sadness_U <- sadness_svd$u
surprise_U <- surprise_svd$u
```

```
# Function to calculate projection and residuals
get_proj_res <- function(candidate, emotion_U) {
    n = ncol(candidate)
    P = emotion_U %*% t(emotion_U)
    proj = P %*% candidate
    res = matrix(0, nrow = 1, ncol = n)
    for (i in 1:n) {
        res[i] = norm((diag(dim(P)[1])-P)%*%candidate[,i])
    }
    return(list(proj, res))
}

anger_image_res <- get_proj_res(test_anger_images, anger_U)[[2]]
contempt_image_res <- get_proj_res(test_contempt_images, contempt_U)[[2]]
disgust_image_res <- get_proj_res(test_disgust_images, disgust_U)[[2]]
fear_image_res <- get_proj_res(test_fear_images, fear_U)[[2]]
happy_image_res <- get_proj_res(test_happy_images, happy_U)[[2]]
sadness_image_res <- get_proj_res(test_sadness_images, sadness_U)[[2]]
surprise_image_res <- get_proj_res(test_surprise_images, surprise_U)[[2]]
```