Facial Recognition with Matrices
CLA365 TECHNICAL REPORT 4

Jeff Wang Justin Potts Jacob Hellenbrand James Bradley

May 2, 2024

Abstract

In this report, we present R-code and images that we used to detect facial recognition between our four group members and outside individuals. Using the SVD of a matrix we compare residual vectors and return the image with the smallest residual as the best match. As well we will present outside sources and their best facial match to each of our four group members.

1 Facial Recognition

1.1 Overview

To create our data set for this analysis we took 30 pictures of each individual of our group. With each individuals images we created 4 matrices, one for each group member. We store each image as a vector/ column of our matrix by gray scaling each image, then vectorizing the matrix once it is gray scaled. We hypothesize that using our test images of each individual, the matrix that is that individuals will have the smallest residual vector when projecting the image into that "J-space".

1.2 Implementation

In terms of the theory, we were heavily inspired by eigenImages and leveraged the techniques to create our person recognizer.

We leverage the SVD to create an orthonormal basis for each matrix of composed of images.

When we perform SVD, we decompose some matrix A into:

- An orthonormal matrix U (the eigenvectors of AA^T).
- A diagonal matrix Σ (diagonal entries are $\sigma_i = \sqrt{\lambda_i}$ such that $A^T A v_i = \lambda_i v_i$ where $1 \geq i \geq rank(A)$).
- And an orthonormal matrix V^T (V is the orthonormal matrix of eigenvectors $v_i \cdots v_r$).

Thus, $A = U\Sigma V^T$.

But how do we create a matrix of images if they are already a matrix? We can work around this by flattening each image of dimension $m \times n$ into a vector of length $m \cdot n$, in other words, each vector in the image matrix is stacked on top of each other to create one image vector, let a single image vector be called f_i.

If we have n images, then our matrix of images can be defined as:

$$F = \begin{bmatrix} | & & | \\ f_1 & \cdots & f_n \\ | & & | \end{bmatrix}$$

We can calculate the mean of the data set $\mu = \frac{1}{n}\sum_{i=1}^{n} f_i$ and create a mean-adjusted matrix of images,

$$F' = \begin{bmatrix} | & & | \\ f'_1 & \cdots & f'_n \\ | & & | \end{bmatrix}$$

Where $f'_i = f_i - \mu$

Once we have our mean-adjusted matrix of flattened images, we perform SVD to get our matrix U, let the number of vectors in this matrix be J, and then we have U matrix we can use for our projection. Our projection is going to be represented as:

$\mathrm{P}_J = \sum_{n=1}^{J} u_i u_i^T$ (1)

Where u_i are the i^{th} vectors of our U matrix.Then, we would project our matrix onto the "J-space" by performing matrix multiplication with the newly found projection matrix and our vectorized test image. Then, we would find the residual by using a property of projections, orthogonal projections, and the identity matrix. The property denotes that the addition of the projection matrix of a space and the orthogonal projection of that same space has to equal the identity matrix. So, to get our orthogonal projection of our "J-space", we have to take the identity and subtract the projection from it, given they both have the dimensions.

2 Testing

2.1 Input Images

For our test images, we had each group member take a picture that would not be used in our training data. From here we will project each members photo onto each of our 4 matricies. Below you can examine the four images we use to test,

(a) (b) (c) (d)

Figure 1: Examine each group members test image: Justin (a), Jacob (b), Jeff (c), James (d)

3 Results

3.1 Using a Test Image

Below we can examine the residuals from project the Jeff test image onto each J-Space matrix. We can observe that Jeff had the smallest residual with himself, showing the Jeff looks most like Jeff.

We can also examine the outputs from Jeff being projected onto each J-Space matrix. It's interesting to observe certain elements can be seen from Jeff's test image, such as his mouth in (d) or his hair in (b).

(a) (b)

(c) (d)

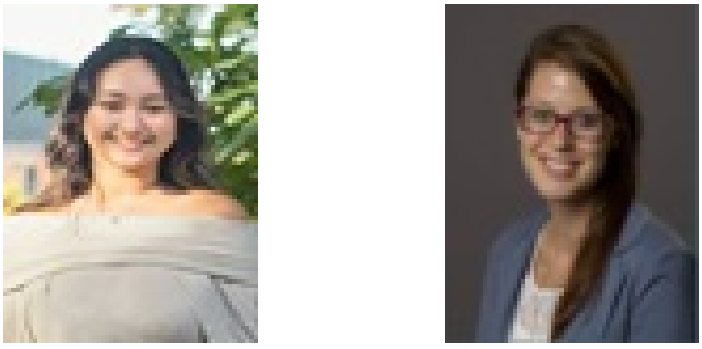
Figure 2: Examine each group member's test image: Jeff (a), Jacob (b), James (c), Justin (d)

J-Matrix	Residual
Jeff	3.224
Jacob	8.251
Justin	10.211
James	15.853

Table 1: A table with the residual amounts for Jeff test image projected onto all 4 J-space matrices

3.2 Outside Examples

Here we will take 2 outside examples and project them onto our J-Space's to see whos face they closet match to. We can observe Figure 3 (a) is our preceptor Carissa, and (b) is our professor Lori.



(a) (b)

Figure 3: Our two test images, (a) our preceptor Carissa, and (b) our professor Lori.

3.3 Did it Work?

Yes! Looking above at our residual table we see that Jeff has the smallest residual associated when projecting his test image onto his matrix. Meanwhile the other 3 residuals are not very close, but we can make the generalizations that Jacob looks second most like Jeff, Justin third most, and James the least.

J Matrix	Lori Residuals	Carissa Residuals
Justin	10.799	15.505
Jacob	11.536	20.981
James	11.777	23.359
Jeff	13.645	26.524

Table 2: A table with four columns

In terms of our outside examples it is a little harder to make a conclusion. Observing the below residual table we see that Lori and Carissa both have the same ordering of people who they closest match to. This could be the result of lighting or better quality images. Sticking true to our functions though we will conclude that Lori and Carissa both look most similar to Justin.

4 Limitations

4.1 Quality and Consistency of Images

One limitation that may have occurred is inconsistency with our images. When taking photos for creating our data we did our best to frame the photos the same to create consistency. With this being said we would say our data is semi consistent due to human error as we did not standardize the way in which we collected our photos.

4.2 Computing Power and Image Reduction

When doing our R-studio calculations we original resized our images to 240×320, hoping this would be a small enough reduction for our computers to be able to compute with. After our first test we were given errors and could not compute at this image size and had to reduce again to 60×80. This is a very large reduction from our original image which was 3024×4032. Below you can examine some data that could be lost in the image size reduction (Note: images would be gray scaled before ran).

(a)

(b)

Figure 4: Here we can see (a) is our original high quality image, compared to (b) which our image after being scaled down to a size our computers can compute.

5 Conclusion

To conclude, after examining our data and projecting our images we feel as if we can accurately predict one of our four group members using our J-Space matricies and the residuals from a projected images. Moving into outside sourced images there may still need to be some photo regulation and function adjustments to best be able to determine which faces are closest matches with our four group members. Overall though our functions can correctly return the best facial match found by the smallest residual.

6 Appendix

```
# Step 1: Convert grey scale images into matrices
```{r}
# library(jpeg)
# ColorImg = readJPEG("HWImages/crabby.jpg")
# img=0.2989*ColorImg[,,1]+0.5870*ColorImg[,,2]+0.1140*ColorImg[,,3]
```

Step 2: Flatten each image into a vector
```{r}
# flattening an mxn matrix into a mnx1 vector
flatten<-function(A){ # A is the grey scale image
  n<-ncol(A)
  v<-c()
  for(i in (1:n)){
    col<-A[,i]
    v<-c(v,col)
  }
  return(v)

}

```

Step 2: Coordinate Tag of the new image
```{r}
# not sure if this will run
coordinate <- function(c,U,j){ # c is our new image
  # U matrix from the svd
  coord <- t(U[,(1:j)]) %*% c
  return (coord)
}
```


Step 3: Create the projection matrix through svd
```{r}
projection <- function(U, j){ # the U matrix from the svd
                              # the first j columns of U
  n <- nrow(U)
  P <- Matrix(0, n, n)
  for (i in (1:j)){
    col <- U[,i]
    P <- P + (col%*%t(col))
  }
  return (P)
}
```

Step 4: Projection Result
```{r}
proj.result <- function(P, coord){
```
```

```
  result <- P %*% coord
  result <- normalize(result)
  return (result)
}
```

```
# Step 5: Residual Result
```{r}
res.result <- function(P, coord){
  I <- diag(ncol(P))
  res <- (I - P)%*% coord
  res <- normalize(res)
  return(res)
}
```

```
## Step 5: Residual Magnitude
```{r}
res.mag <- function(r){ # r is the residual
  return (vnorm(r))
}
```

```
# Step 6: Combine all of the previous functions together
```{r}
ExpressionCorrelation <- function(A, c, j){ # A is the matrix that contains all of vectorized pictures
                                            # c is our image that we are testing it against
  mu <- A$mu
  A <- A$M

  c.vector <- c(c) # the j dimensions we want from our U matrix

  u <- svd(A)$u
  p <- projection(u,j)
  result <- proj.result(p, c.vector)
  res <- res.result(p, c.vector)

  # plot the original image
  imPlot(c, main = 'original image')

  # plot the projected image
  result<-as(array(result,dim = dim(c)),"matrix")
  imPlot(result, main = 'projected image')

  # plot the residual of the image
  res1<-as(array(res,dim = dim(c)),"matrix")
  imPlot(res1, main = 'residual image')
  res.magnitude <- res.mag(res)

  return (res.magnitude)
}
```
```

```
# Making the matrices
ExpressionMatrix<-function(path){
  pics<- list.files(path, pattern ="*.jpeg", ignore.case = TRUE, full.names = TRUE)
  n<-length(pics)
  M<-c()
  # loop through all of the images in a directory
  for(i in 1:n){
    imgs<-pics[i]
    real.img<-readJPEG(imgs)
    gray.img<-gray.scale(real.img)
    # we use t() here because we flatten each image by row first
    vec<-c(gray.img)
    M<-cbind(M,vec)
  }

  # find the mu value and subtract from M
  mu<-rowMeans(M)
  #M <- M-mu

  return(list(M = M, mu = mu))
}
```