Face Recognition with Holistic methods

Chenze Li

March 18, 2024

1 Background

The application and techniques of face recognition have a long history and are becoming more intelligent and accessible with the development of computer science and the increase in volumes of data that people can process. As early as 1871, people compared a part of a facial photograph to identify a person at a British court [AOBTA20]. Bledsoe et al. proposed a method of facial recognition where twenty measures such as the size of mouths or eyes are asked to be input. Later the first successful facial recognition technique, Eigenfaces, appeared in 1991 by Alex Pentland and Matthew Turk[AOBTA20]. Recently, Deepface developed by Facebook in 2014 was said to have achieved 97% of the performance of human eyes [AOBTA20].

The techniques of face recognition have been requested in areas of security, law enforcement, and many commercial fields because of some favored characteristics. First of all, the face is the most common and direct biological feature to identify people. Besides, facial information is easier to collect and access without physical contact compared with iris and fingerprints. Finally, the collection of facial information requires little assistance from data collectors. [AOBTA20] With all these advantages, it is essential to develop face recognition.

With several decades years of growth, many face recognition systems are proposed and can be classified into four kinds of approaches which are holistic methods, local (geometrical) methods, local texture descriptors-based methods, and deep learning-based methods [AOBTA20]. This report focuses on holistic methods. Typically, holistic methods align graph matrices into vectors and project vectors into lower dimension spaces [KJAFA20]. Then for clarification, a new facial image will be projected into that subspace and the distances from the new image vector to all classes in the subspace will be calculated. Such techniques include principal component analysis(PCA), also known as eigenface, linear discriminative analysis (LDA), a.k.a. fisherfaces, and independent component analysis (ICA)[KJAFA20].

In this report, the image data contains facial images of the same group of people and is divided into a training set and a test set. The goal is to train the images in the training set with holistic methods and identify the same people with the new images from the test set. Each image in the data is a $s_1 \times s_2$ matrix and has been aligned into a $s_1 \times s_2$ vector. There are n_2 images for n_1 people in the training set and therefore the training set is a $(s_1 \times s_2) \times (n_1 \times n_2)$ matrix. The first n_2 columns represent images of person 1, the second $n2$ columns represent images of person 2, and so on. The test set is arranged in a similar form to the training set.

2 Methodology

The general idea of the following techniques is to find a projection matrix U_1 and project the training set Y_{train} onto that subspace and classify the images in a low-dimensional subspace that describes the data economically. That is,

$$Y_1 = U_1^\top Y_{train}$$

Project a newly captured image I which is a $s_1 \times s_2$ vector onto that subspace, that is,

$$I_1 = U_1^\top I$$

and calculate the L_2 norm distances from the new image to each image in the projected training set and the predicted class for the new image is the class with the smallest distance. Assume the vector corresponding to j-th image of person i in the training set as Y_{ij} and denote $\hat{Y}_{ij}$ as the vector after projection, that is,

$$\hat{Y}_{ij} = U_1^\top Y_{ij}$$

Then I is predicted as the image of person m if

$$m = \underset{i=1,2,\cdots,n_1}{\operatorname{argmin}} \|\hat{Y}_{ij} - I_1\|_2, \quad \forall j = 1, 2, \cdots, n_2$$

The following shows different methods to find such an optimal projection matrix which somehow captures as much information as possible.

2.1 Principle Component Analysis

The main idea for PCA is to minimize the mean squared error and leave the features with the largest variance. If the dimension of the subspace is expected to be k, the traditional method to find a projection matrix is to use the first k columns of the orthogonal matrix in the singular value decomposition(SVD) of the covariance matrix of the data matrix. However, it is expensive to calculate the covariance of the original matrix. Therefore, SVD is directly applied to the training set. Assume the training set is denoted as Y_{train} and

$$Y_{train} = U\Sigma V^\top$$

U_1 is the first k columns of U and is used as the projection matrix.

2.2 Fisher Discriminative Analysis

Fisher discriminative analysis(FDA) shows another method to generate the projection matrix. It starts from the between-class scatter matrix denoted as S_B and the within-class scatter matrix denoted as S_W. Obviously, the observations within the same class should be closer and the observations from different classes should be far enough. That is, the variance among observations within the same class should be minimized whereas the variance among observations across classes should be maximized. Then the goal of FDA is to find a projection matrix to minimize the within-class variance and maximize the between-class variance simultaneously.

Assume the data set has m labels and n_i observations for label i. For example, $X_{ij} \in \mathbb{R}^n$ is the j-th observation for label i. $\bar{X}_i$ is the sample mean for label i and $\bar{X}$ is the sample mean of all the observations. $U_1 \in \mathbb{R}^{n\times d}$ is the projection matrix. Denote Z_{ij} as the data point after projection, that is,

$$Z_{ij} = U^\top X_{ij}.$$

The definitions of S_B and S_W are

$$S_B = \sum_{i=1}^{m} \left(\bar{X}_i - \bar{X}\right)\left(\bar{X}_i - \bar{X}\right)^\top$$

$$S_W = \sum_{i=1}^{m}\sum_{j=1}^{n_i} \left(X_{ij} - \bar{X}_i\right)\left(X_{ij} - \bar{X}_i\right)^\top$$

Then the between-class scatter matrix S_B^Z for projected observations and the within-class scatter matrix S_W^Z are shown below:

$$\begin{aligned}
S_B^Z &= \sum_{i=1}^{m} \left(\bar{Z}_i - \bar{Z}\right)\left(\bar{Z}_i - \bar{Z}\right)^\top \\
&= \sum_{i=1}^{m} \left(U^\top\bar{X}_i - U^\top\bar{X}\right)\left(U^\top\bar{X}_i - U^\top\bar{X}\right)^\top \\
&= \sum_{i=1}^{m} U^\top\left(\bar{X}_i - \bar{X}\right)\left(\bar{X}_i - \bar{X}\right)^\top U \\
&= U^\top S_B U
\end{aligned}$$

$$
\begin{aligned}
S_W^Z &= \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(Z_{ij}-\bar{Z}_i\right)\left(Z_{ij}-\bar{Z}_i\right)^\top \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n_i}\left(U^\top X_{ij}-U^\top \bar{X}_i\right)\left(U^\top X_{ij}-U^\top \bar{X}_i\right)^\top \\
&= \sum_{i=1}^{m}\sum_{j=1}^{n_i} U^\top\left(X_{ij}-\bar{X}_i\right)\left(X_{ij}-\bar{X}_i\right)^\top U \\
&= U^\top S_W U
\end{aligned}
$$

Then the projection matrix should minimize the within-class variance and maximize the between-class variance, that is,

$$
U = \underset{U\in\mathbb{R}^{n\times d}, U^\top U=I_d}{\operatorname{argmax}} \frac{det\left(U^\top S_B^Z U\right)}{det\left(U^\top S_W^Z U\right)}
$$

The solution to that maximization problem is related to the generalized eigenvectors shown below:

$$
S_B x_\lambda = \lambda S_W x_\lambda
$$

The projection matrix U is the eigenvectors corresponding to d largest generalized eigenvalues.

Combined with the setting of the problem in the report, apply PCA to the training set at first reducing the dimension from $(s_1 \times s_2) \times (n_1 \times n_2)$ to $\frac{(n_1\times n_2)}{2} \times (n_1 \times n_2)$, that is,

$$
\widetilde{Y}_{train} = U_0^\top Y_{train}
$$

where U_0 is the projection matrix from PCA. The FDA is applied to the new training matrix $\widetilde{Y}_{train}$. The first n_2 columns represent the images of the first person and so on. Then the projection matrix for $\widetilde{Y}_{train}$ is obtained through the previously introduced process which is denoted as V. Finally, combined with PCA, the projection matrix U_1 starting from Y_{train} is naturally

$$
U_1 = U_0 V.
$$

which reduces the dimension of Y_{train} from $(s_1 \times s_2) \times (n_1 \times n_2)$ to $k \times (n_1 \times n_2)$.

2.3 Simple Projection

The most naive method to generate the projection matrix is to select the first k columns of an identity matrix, that is,

$$
U_1 = (e_1, \cdots, e_k)
$$

where e_i is a $(s_1 \times s_2)$ vector with i-th position as 1 and others as 0.

3 Results

The sizes of the data set used in the report are following:

$$
s_1 = 28, s_2 = 23, n_1 = 40, n_2 = 5
$$

The performances of the three methods are evaluated by correct classification rate. The dimension of the subspace k is also changed to see the trend of the rates. As shown in Figure 1, with the increase of the dimension of the subspace, the accuracy increases which follows our intuitions. Another interesting phenomenon is that the performance of FDA is not always better than that of PCA although FDA performs better at most times. This happens when the dimension is small. Besides that, the accuracy does not always increase with the increase of the dimension and sometimes the accuracy will drop.

Some test images and the closest images in the training set will be shown. I select some images when $k = 1, 9, 10, 20, 30, 40$. Some of them are special points. For example, when $k = 1$, the accuracy of simple projection is the largest among the three methods, and when $k = 9, 10$, the correct classification rate of PCA is larger than that of FDA. The first image of the person 15, the third image of the person 30 and the fifth image of person 40 in the test set will be selected to be shown. In Figure 2, 3 and 4, the images on the left column are the original selected images in the test set, and the six images on the right side are the corresponding closest images in the training set. The numbers shown above figures are the predicted labels.

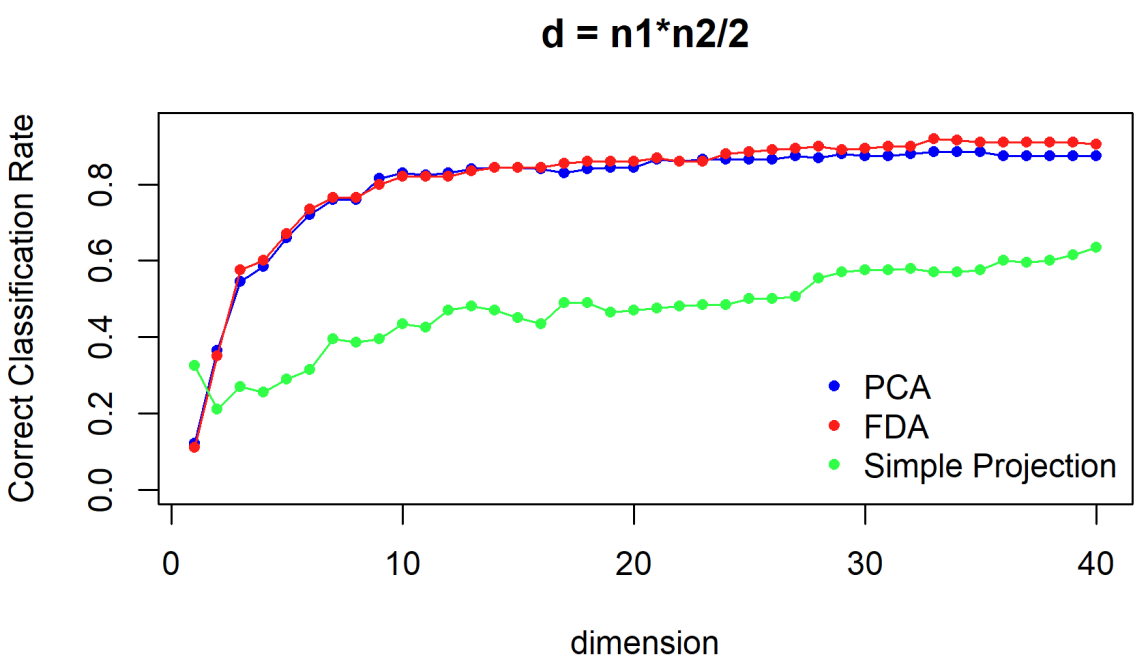

Figure 1: Correct Classification Rate with different dimensions

Dimension	1	2	3	4	5	6	7	8
PCA	12.00%	36.50%	54.50%	58.50%	66.00%	72.00%	76.00%	76.00%
FDA	11.00%	35.00%	57.50%	60.00%	67.00%	73.50%	76.50%	76.50%
Simple Projection	32.50%	21.00%	27.00%	25.50%	29.00%	31.50%	39.50%	38.50%
Dimension	9	10	11	12	13	14	15	16
PCA	81.50%	83.00%	82.50%	83.00%	84.00%	84.50%	84.50%	84.00%
FDA	80.00%	82.00%	82.00%	82.00%	83.50%	84.50%	84.50%	84.50%
Simple Projection	39.50%	43.50%	42.50%	47.00%	48.00%	47.00%	45.00%	43.50%
Dimension	17	18	19	20	21	22	23	24
PCA	83.00%	84.00%	84.50%	84.50%	86.50%	86.00%	86.50%	86.50%
FDA	85.50%	86.00%	86.00%	86.00%	87.00%	86.00%	86.00%	88.00%
Simple Projection	49.00%	49.00%	46.50%	47.00%	47.50%	48.00%	48.50%	48.50%
Dimension	25	26	27	28	29	30	31	32
PCA	86.50%	86.50%	87.50%	87.00%	88.00%	87.50%	87.50%	88.00%
FDA	88.50%	89.00%	89.50%	90.00%	89.00%	89.50%	90.00%	90.00%
Simple Projection	50.00%	50.00%	50.50%	55.50%	57.00%	57.50%	57.50%	58.00%
Dimension	33	34	35	36	37	38	39	40
PCA	88.50%	88.50%	88.50%	87.50%	87.50%	87.50%	87.50%	87.50%
FDA	92.00%	91.50%	91.00%	91.00%	91.00%	91.00%	91.00%	90.50%
Simple Projection	57.00%	57.00%	57.50%	60.00%	59.50%	60.00%	61.50%	63.50%

Table 1: Correct Classification Rate with different dimensions

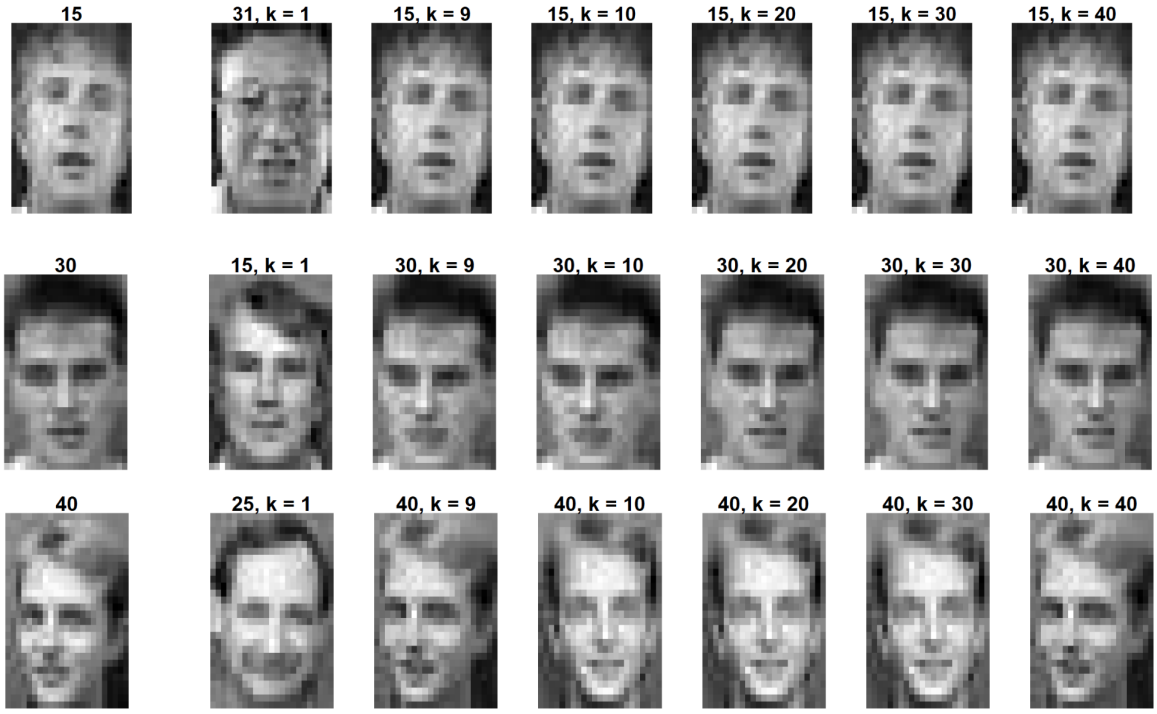

Figure 2: PCA

Figure 3: FDA

Figure 4: Simple Projection

4 Discussion

As shown in Figure 1 and table 1, it is easy to summarize that FDA performs the best among the three methods and PCA performs not as well as FDA but still has a high accuracy. The simple projection performs the worst. Apart from that, the trend of simple projection is very stable with the increase of dimensions but for PCA and FDA, the increase is very sharp when the dimension is low and then becomes stable when the dimension is large.

In summary, FDA is more complicated and takes more time compared with PCA but the performances of these two methods are similar. In my opinion, PCA is recommended among these three methods. However, other problems should be taken into account in the application. For example, the facial images may not be so standard, and some ambient conditions such as lightning, facial expressions, angles, and movement of the camera may change and affect the performances of these methods. Therefore, holistic methods may not satisfy our needs and other nonlinear techniques or more advanced methods such as deep learning and convolutional neural networks may be more suitable in more complicated situations.

After the method is decided, the dimension of the subspace needs to be carefully considered. From Figure 1, it is obvious that after the dimension exceeds 10, the speed of the increase in terms of accuracy slows down. Therefore, to balance the computational complication and accuracy of the model, the dimension around 10 is recommended.

References

[AOBTA20] Insaf Adjabi, Abdeldjalil Ouahabi, Amir Benzaoui, and Abdelmalik Taleb-Ahmed. Past, present, and future of face recognition: A review. *Electronics*, 9(8):1188, 2020.

[KJAFA20] Yassin Kortli, Maher Jridi, Ayman Al Falou, and Mohamed Atri. Face recognition systems: A survey. *Sensors*, 20(2):342, 2020.

R Programming

```
### import data

train = read.table("traindata.txt")
test = read.table("testdata.txt")
train = as.matrix(train)
test = as.matrix(test)


n1 = 40
n2 = 5
s1 = 28
s2 = 23
# k = 10
U = svd(train)$u
# PCA

PCA<-function(s){


  U1_P = U[,1:s]
  return (U1_P)

}


# FDA

library(geigen)
library(far)
library(nlme)

d = n1*n2/2
U_d = U[,1:d]
Y_train = t(U_d)%*%train
Y_mean = as.matrix(apply(Y_train,1,mean))
Sb = 0
Sw = 0

for (i in 1:n1){
  Y_se = as.matrix(Y_train[,((i-1)*n2+1):(i*n2)])
  Y_bet = as.matrix(apply(Y_se,1,mean))
  Sb = Sb + (Y_bet - Y_mean)%*%t(Y_bet - Y_mean)
  for (j in 1:n2){
    Sw = as.matrix(Sw + (Y_se[,j]-Y_bet)%*%t(Y_se[,j]-Y_bet))
  }

}

Sb =  as.matrix(Sb)
Sw = as.matrix(Sw)
result = geigen(Sb,Sw)

FDA<- function(s){
  V = result$vectors[,(d-s+1):d]
  V = orthonormalization(V,basis = FALSE, norm = TRUE)
  U1_F = U_d%*%V

  return (U1_F)
}
```

```
# Simple Projection
SIP<-function(s){
  I = diag(1,nrow = s1*s2, ncol = s1*s2)
  U1_S = I[,1:s]
  return (U1_S)
}

# calculate the accuracy

match<-function(X,Y){
  n = dim(Y)[2]
  dis = c()
  for (i in 1:n){
    dis = c(dis,norm(X - Y[,i],type = "2"))
  }
  ind = which(dis == min(dis))
  return (unique(ceiling(ind/n2)))
}


classification<-function(U1,I){
  k = 0
  pre = 0
  Y = t(U1)%*%train
  I1 = t(U1)%*%I
  n = dim(I1)[2]
  for (i in 1:n){
    I_i = I1[,i]
    pre = match(I_i,Y)
    if (ceiling(i/n2) %in% pre){
      k = k+1
    }
  }
  return (k/n)
}

pca = c()
fda = c()
sip = c()
k_n = 40

for (k in 1:k_n){
  pca = c(pca, classification(PCA(k),test))
  fda = c(fda, classification(FDA(k),test))
  sip = c(sip, classification(SIP(k),test))
}

# plot the trend of accuracy with the increase of dimension

plot(1:k_n,pca,type = "o",pch = 20,xlab = "dimension",
ylab = "Correct␣Classification␣Rate",main = "d␣=␣n1*n2/2",
ylim = c(0,0.95),col = "blue")
lines(1:k_n, fda, type = "o", pch = 20,col = "red")
lines(1:k_n, sip, type = "o", pch = 20,col = "green")
legend("bottomright",pch = c(20,20),legend = c("PCA","FDA",
"Simple␣Projection"), col = c("blue","red","green"),bty = "n")

# show some examples of classification

newmatch<-function(X,Y){
  n = dim(Y)[2]
```

```
  dis = c()
  for (i in 1:n){
    dis = c(dis,norm(X - Y[,i],type = "2"))
  }
  ind = which(dis == min(dis))
  return (ind)
}
toys<-function(U1, I){
  Y = t(U1)%*%train
  I1 = t(U1)%*%I

  pre = newmatch(I1,Y)

  return (pre)
}

# select some images in the test set

ex_k = c(1,9,10,20,30,40)
ex_i = c(15, 30, 40)
ex_j = c(1,3,5)
im_se = (ex_i - 1)*5+ex_j

### PCA
par(mar=c(1,1,1,1),oma=c(1,1,1,1))
for (i in im_se){
  layout(matrix(c(1,0,2,3,4,5,6,7), ncol=8),
  widths = c(2,0.5, 2,2,2,2,2,2))
  s = test[,i]
  m = matrix(s, nrow = s1, ncol = s2)
  m = apply(m, 2, rev)
  image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
  title(main = as.character(ceiling(i/n2)))
  for (j in ex_k){
    p = toys(PCA(j),s)
    imag_ = train[,p]
    m = matrix(imag_, nrow = s1, ncol = s2)
    m = apply(m, 2, rev)
    image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
    title(main = paste0(as.character(ceiling(p[1]/n2)),
    paste0(",␣k␣=␣",as.character(j))))
  }
  layout(matrix(1))
}


### FDA
par(mar=c(1,1,1,1),oma=c(1,1,1,1))
for (i in im_se){
  layout(matrix(c(1,0,2,3,4,5,6,7), ncol=8),widths = c(2, 0.5, 2,2,2,2,2,2))
  s = test[,i]
  m = matrix(s, nrow = s1, ncol = s2)
  m = apply(m, 2, rev)
  image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
  title(main = as.character(ceiling(i/n2)))
  for (j in ex_k){
    p = toys(FDA(j),s)
    imag_ = train[,p]
    m = matrix(imag_, nrow = s1, ncol = s2)
    m = apply(m, 2, rev)
    image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
    title(main = paste0(as.character(ceiling(p[1]/n2)),
```

```
    paste0(", k = ",as.character(j))))
  }
  layout(matrix(1))
}

### Simple Projection
par(mar=c(1,1,1,1),oma=c(1,1,1,1))
for (i in im_se){
  layout(matrix(c(1,0,2,3,4,5,6,7), ncol=8),widths = c(2, 0.5, 2,2,2,2,2,2))
  s = test[,i]
  m = matrix(s, nrow = s1, ncol = s2)
  m = apply(m, 2, rev)
  image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
  title(main = as.character(ceiling(i/n2)))
  for (j in ex_k){
    p = toys(SIP(j),s)
    imag_ = train[,p[1]]
    m = matrix(imag_, nrow = s1, ncol = s2)
    m = apply(m, 2, rev)
    image(t(m),axes = FALSE,col = grey(seq(0, 1, length = 256)))
    title(main = paste0(as.character(ceiling(p[1]/n2)),
    paste0(", k = ",as.character(j))))

  }
  layout(matrix(1))
}
```