STA5106 Midterm

Facial Recognition Methods

FSU Statistics Department

Table of Contents

Problem:

Facial recognition is a technology that, while it has been around for years, has never really been perfected (at least not to the knowledge of the public). Presumably, there are too many factors at play to get it just right – changes in lighting, changes in a person's facial hair, different facial expressions at the time of scanning, etc. To simplify this issue in the case of this project, images of 200 people have been decolorized and down-sampled to allow us students to look at the differences between the performances of two popular methods of feature extraction using two different methods of dimension reduction – namely Principal Component Analysis and Simple Projection. The purpose of this report is to examine the differences in accuracy and efficiency between these two methods. To do this, we are given two data sets of facial images, Y_{train} and Y_{test}, and the objective is to see how effective each method is at recognizing different images of the same people.

Methodology:

Let s_1 x s_2 be the size of each individual image provided in Y_{train} (28 x 23 in the case of this project).

Simple Projection: Define U_1 to be the first k columns of the s_1 x s_2 identity matrix. Then, using the projection $Y_1 = U_1{}^{\text{T}} Y_{\text{train}}$, each image in Y_{train} can be reduced to a k-dimensional vector.

Principal Component Analysis: Using Singular Value Decomposition, derive the orthogonal matrix U. Let U_1 in this case be the first k columns of the orthogonal matrix U, then use the same projection $Y_1 = U_1{}^{\text{T}} Y_{\text{train}}$, to reduce each image in Y_{train} to a k-dimensional vector.

Nearest Neighbor Classification: Let I represent a single image from Y_{test}. Let the projection of I be $I_1 = U_1{}^{\text{T}} I$. Compute the distance between the I_1 that has just been calculated, and every column of Y_1. The label of the column that has the smallest distance to I_1 should match the true label. If so, it is a successful match, if not it is a failure.

The two methods of facial recognition we are comparing are as follows:

Method 1: Simple Projection -> Nearest Neighbor Classification
Method 2: Principal Component Analysis -> Nearest Neighbor Classification

For k = 1,2,…,40, the percentage of images that are correctly identified will then be calculated (notated by *F(k)*), plotted and compared for each method. The efficiency ratios at accuracies above 0.8 (defined in the Results section) for four runs of the program will also be calculated, plotted, and compared.

Programs:

The implementation of each of these programs applies the methodology that is outlined above, but with for-loops worked into the program in order to iterate through all of the images included in the provided data sets. For example, the program for Method 2 includes a set of nested for-loops - the innermost of which implements the Nearest Neighbor Classification step for each given k, and the outermost of which applies Principal Component Analysis to Y_{train} and calculates accuracy for $k = 1,\ldots,40$. The program for Method 1 works similarly, but obviously applies Simple Projection in the outermost loop rather than Principal Component Analysis.

In addition to the programs that implement the two facial recognition methods, code is included to plot k vs. $F(k)$, as well as computation time at each $k = 1,\ldots,40$ in order to compare the accuracy of each of these methods. Code to see whether or not each method was successful in recognizing specific instances from Y_{test} is included as well. Finally, code that calculates the Efficiency Ratio between the two methods and plots them against 13 different accuracies is included at the end.

Results:

Accuracy:

In terms of accuracy, Method 2 (PCA) performed significantly better than Method 1 (Simple Projection) at every $k = 2,\ldots,40$, as we can see below:

Figure 1: Accuracy Comparison for $k = 1,\ldots,40$

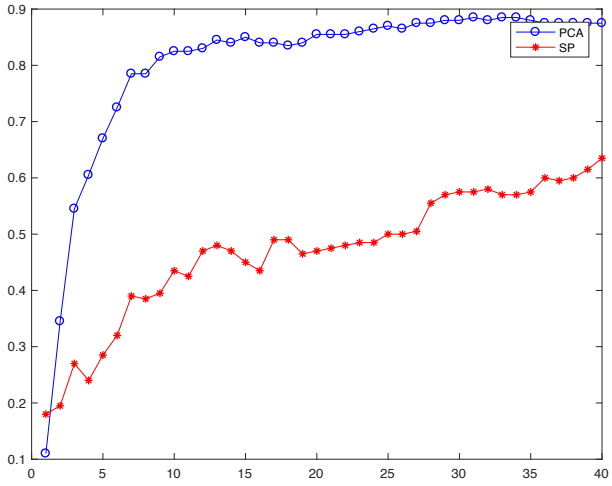
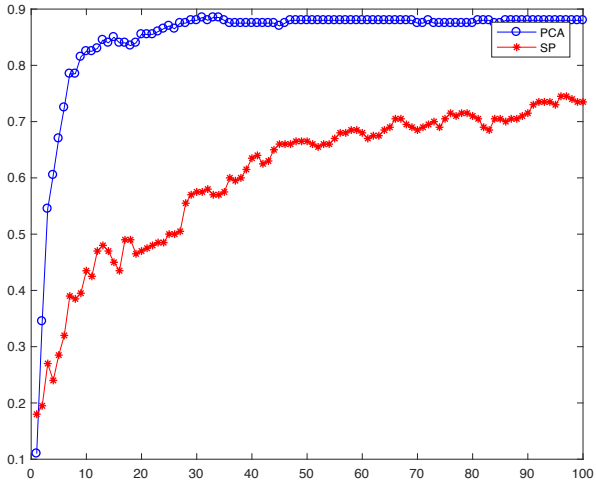


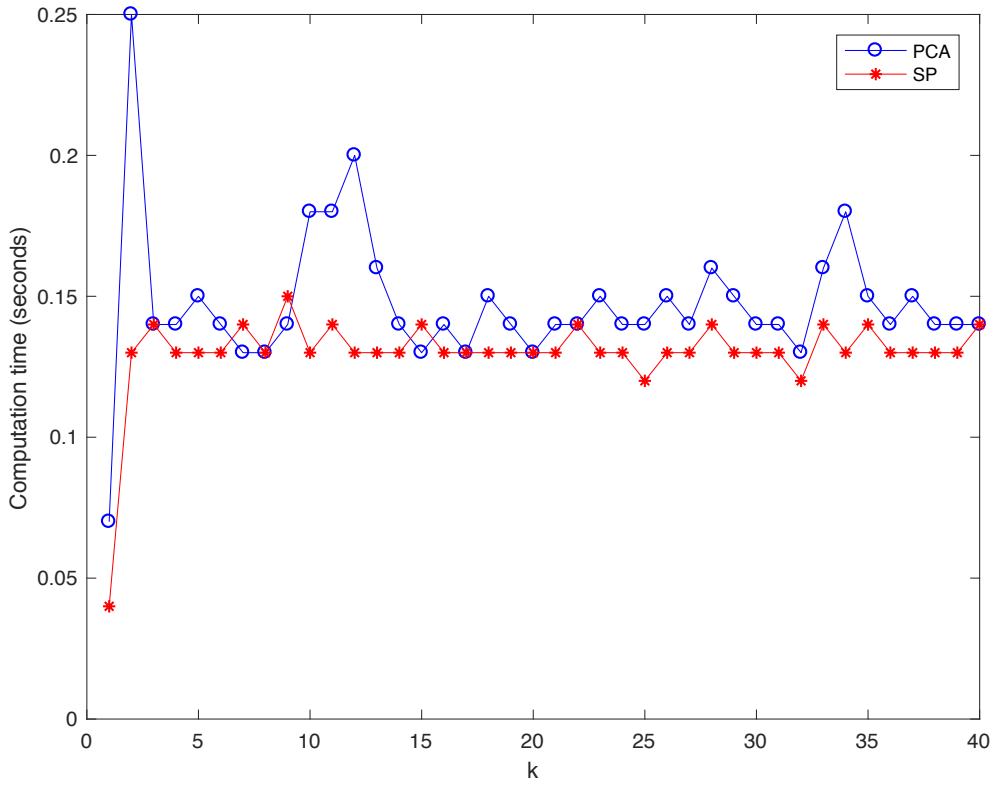
Figure 2: Accuracy Comparison for $k = 1,\ldots,100$



In *Fig. 1* and *Fig. 2*, the x-axis represents each k, while the y-axis represents the percentage of test images that were correctly identified. *Fig. 1* shows the accuracies for $k = 1,\ldots,40$, while *Fig. 2* represents the accuracies for $k = 1,\ldots,100$. It is clear from the plots that even at $k = 100$, Method 2 performs significantly better than Method 1. Furthermore, it may be worthy to note that Method 2 seems to reach a ceiling in terms of accuracy around $k = 30$, while the trend in accuracy for Method 1 appears to be steadily increasing even at $k = 100$. Looking at the behavior of Method 1, one may wonder if it ever reaches comparable levels of accuracy to Method 2, and if so, at what point? This brings to mind a question of not just accuracy, but efficiency.

Remarks on Efficiency:

Figure 3: Computation Time Comparison for k = 1,…,40



As can be seen in *Fig.* 3, from k = 1,…,40, Method 1, utilizing Simple Projection, generally seems to be faster than Method 2, utilizing Principal Component Analysis. Additionally, when running the program iteratively, the "tic toc" code showed that for most k's, the time elapsed after iterating through Method 1, was typically less than the time elapsed after iterating through Method 2.

However, as discussed in the Accuracy section, Method 1, although faster than Method 2 at most k's, is also less accurate. This begs the question: which method is more efficient? Another way to think about this question is, which method requires the least computational cost to get the most accurate result? After running the programs at the maximum possible value of k (k = 644), I used the max() function to find the highest accuracy that each method achieved, the corresponding k, and the associated computational time required to achieve that accuracy (bear in mind, computational time for the corresponding k^{th} loop is going to be significantly lower than the cumulative time it takes for the program to run through all iterations):

	Maximum Accuracy	***k***	**Computational Time**
Method 1	0.880	644	0.34s
Method 2	0.885	31	0.15s

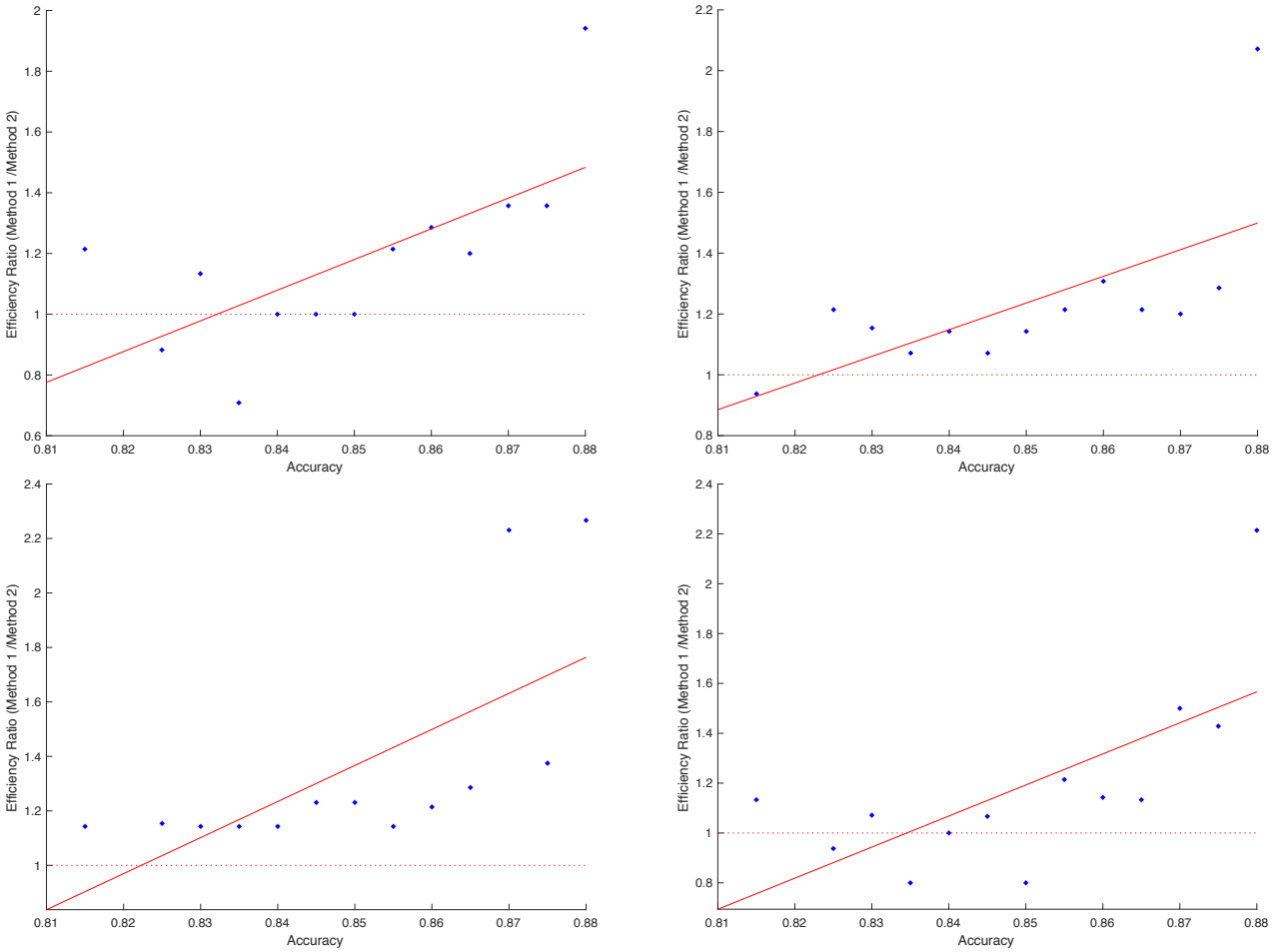
From here we can create a ratio of the computational times to see that for Method 1 to reach it's maximum accuracy it takes approximately 0.34/0.15 = 2.27 times as long as Method 2. However, we can see that this definition led us to an extreme example, where it just happened to take Method 1 the maximum number of iterations to reach it's maximum accuracy. Suppose we were willing to sacrifice half a percentage point of accuracy to save some computational time with Method 1. We then get the following chart:

	Accuracy	***k***	**Computational Time**
Method 1	0.875	349	0.21s
Method 2	0.875	27	0.14s

We see here that for both methods to reach an accuracy of 0.875 (only 0.01 less accurate than the maximum accuracy of Method 2), it takes Method 1 only 1.5 times as long as Method 2, which is a significant increase in efficiency for Method 1, with only a slight decrease in accuracy.

Efficiency, therefore, is something that varies greatly depending on the threshold for accuracy that is required. Not only that, but it also depends on the specific instance that the program is run. Different runs of the program lead to different computation times for different accuracies. Because of the drastic effect on computation time that these two variables have, I decided to graph the efficiency ratios (Computation time for Method 1 to reach a specific accuracy/ Computation time for Method 2 to reach a specific accuracy) at several different accuracies, for four separate runs of the program. I decided to only graph accuracies above 0.8 in order to retain some relevance to practical use of this facial recognition technique (I can't imagine it would be useful in practice if the accuracy was below 0.8):

Figure 4: Efficiency ratios for accuracy thresholds above 0.8 for four separate program runs



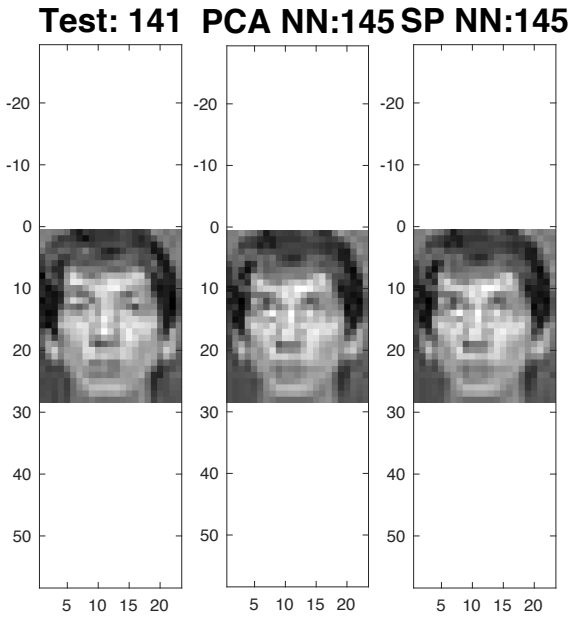
There are two major conclusions we can draw from *Fig. 4*. Despite the fact that we are only looking at $n = 4$ separate runs of this program, the overall trend on each plot points toward the following:

1. Because the majority of points on each of these plots are greater than 1, the computation time required to reach a particular accuracy threshold above 0.8 is usually greater for Method 1 than for Method 2.

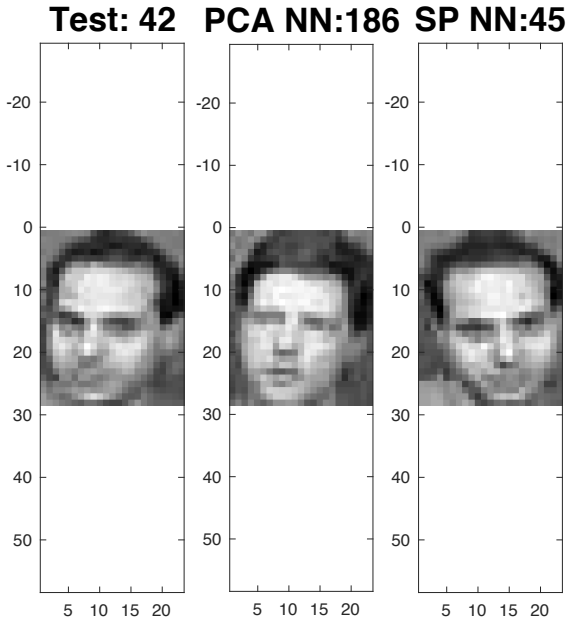
2. After fitting a least-squares regression line to the points, it is clear that the efficiency ratio in general becomes larger as the accuracy threshold increases. In other words, the magnitude by which Method 2 is faster than Method 1, increases as the required accuracy threshold increases.

Examples:

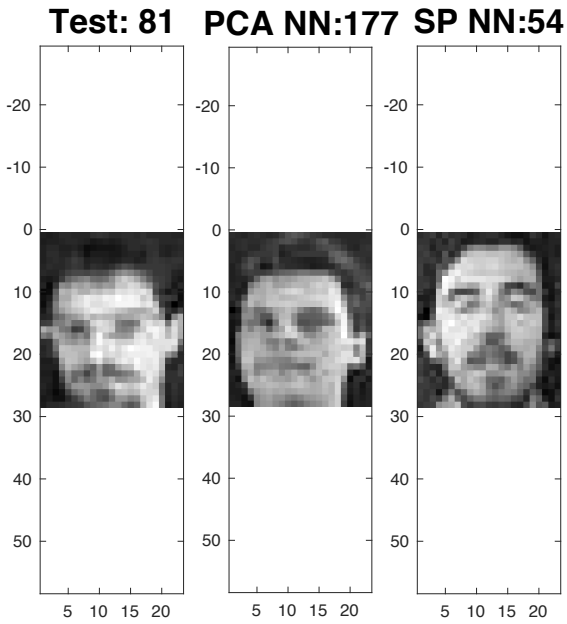
Figure 5: Examples of correctly and incorrectly identified faces for Method 1 vs. Method 2 Comparison



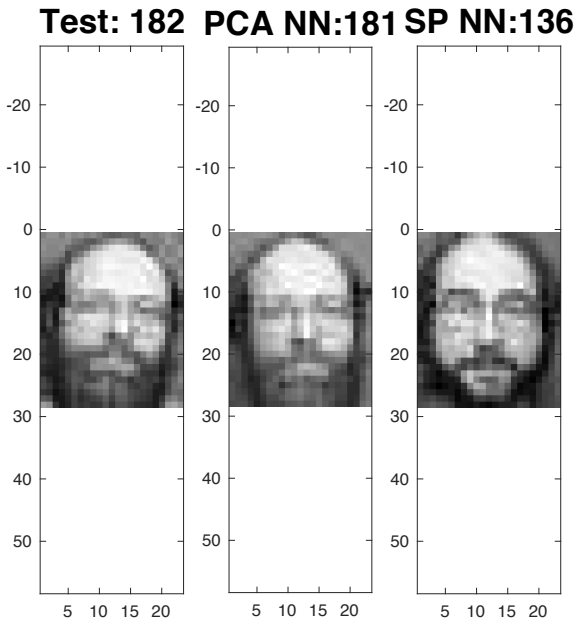
Method 1: Correct Method 2: Correct



Method 1: Correct Method 2: Incorrect



Method 1: Incorrect Method 2: Incorrect



Method 1: Incorrect Method 2: Correct

Summary:

Based off of the information gathered with this project, we can make a number of conclusions about the accuracy and efficiency of facial recognition using Simple Projection vs. facial recognition using Principal Component Analysis. In terms of accuracy, based off of *Figs. 1* and *2*, it is clear that Method 2 is more accurate than Method 1 for at least every $k = 1,\ldots,100$. However in practical usage, efficiency is also a very strong factor to consider when deciding between these two methods.

Looking at *Fig. 4*, and taking into account the two conclusions made after it's presentation, there is evidence to support Method 2 being the more efficient facial recognition method, especially at higher accuracy thresholds. It is worth noting however, the program was only run $n = 4$ times. This is not enough repetition to draw any concrete conclusions. Further efficiency analysis for this problem could involve more repetition and even statistical analysis of the efficiency ratios to see if there is sufficient evidence to make the claim that the true value of the efficiency ratio is greater than 1.

It is also worth noting that at the computation times that occurred in this project, the differences in times between methods may seem negligible. After all, most iterations for both methods took less than 0.35 seconds each to run. In practical usage, one may make the case that at these speeds, it doesn't really matter which is faster. However, in actual facial recognition software, not only are the dimensions of the images much higher, but a higher threshold of accuracy is required. This means that the conclusions made on the efficiency of Method 1 vs. Method 2 in this report should theoretically remain the same in actual facial recognition programs, where the times become much higher and the differences become a matter of seconds, not fractions of seconds.

Finally, although Method 2 appears to be superior to Method 1 in terms of accuracy and efficiency, there are certainly limitations to this method. *Fig. 5*, for example, shows an instance where Method 2 incorrectly identified a face, while Method 1 correctly identified the same face. This example is important because it makes clear an important aspect of the interpretation of the accuracies of each method. That is, just because Method 2 performed with a higher accuracy at a given k, does not mean that it correctly identified every face that Method 1 did. Even though Method 1 underperformed, there could still be cases where Method 1 correctly identifies faces that Method 2 does not. Further study may consist of finding out which types of faces Method 1 is more accurate at identifying, in order to combine it with the Method 2 to create an even better facial recognition algorithm.

Extension Code:

```
C = unique(acc_pca,'first')
C = C(C>0.8)
C = C(C<0.885)

Ck_pca = zeros(size(C))
Ck_sp = zeros(size(C))

for i = 1:length(C)
    Ck_pca(i) = find(acc_pca == C(i),1)
    Ck_sp(i) = find(acc_sp == C(i),1)
end

EffRatio = zeros(size(C))

for j = 1:length(C)
    EffRatio(j) = e2(Ck_sp(j))/e1(Ck_pca(j))
end

 plot1 = scatter(C, EffRatio,10,'b*');
 xlabel('Accuracy');
 ylabel('Efficiency Ratio (Method 1 /Method 2)')
 hold on;
 h = lsline;
 set(h,'color','r');
 hold on;
 g = hline(1);
```