FACIAL RECOGNITION ALGORITHMS

COREY COCHRAN-LEPIZ
HWEI-SHIN HARRIMAN

1. Abstract

Humans are remarkably skilled at identifying faces; we associate faces to our friends, family, supervisors, celebrities, and acquaintances with little to no effort. Developing a facial recognition program that is nearly as robust, accurate, and fast as a human presents many challenges. For example, superfluous variation in datasets of faces can easily confuse a computer. Our goal in this paper is to minimize needless variation while emphasizing data that is important for facial recognition. We do this by applying two facial recognition algorithms–the Euclidean-distance-driven Eigenfaces and the probabilistic Bayesian approach–to a data set of faces taken from our class. We break down the mathematical concepts of each algorithm, including Singular Value Decomposition, Principle Component Analysis, and Bayesian normal distribution. Our results showed that the Eigenface algorithm was more effective than the Bayesian algorithm, with the former being 90.16% accurate and running in 0.000001 seconds and the latter being 86.89% accurate and running in 0.00007 seconds. While further investigation is necessary, the results point towards Bayesian for real-world application.

2. Introduction

The human face is very complex and nuanced. We, as humans, are extremely skilled at picking up minor differences in faces without much effort. An interesting challenge to consider is how humans can train computers to recognize faces. Computer vision facial recognition algorithms have a number of practical applications in security, social media, and identification. Because of their usefulness, many algorithms have been developed in an attempt to find the fastest, most accurate, and most robust method of implementation. There are some common issues that each method must address in order to be functional, such as changes in appearance of the person, environment, scale, angle, and quality of the image, or the fact that many people have similar complexions, eye colors, and features. Therefore, analyzing a face with equal emphasis on every part of the image will not be effective.

Date: April 2018.

One method to address some of these issues is to minimize the variation between face images (i.e setting the camera at a uniform distance, setting a uniform brightness and file size, taking pictures head-on, centering the face and making the face the primary focus of the image, etc.). These steps will minimize the irrelevant variation that can easily inhibit the computer's recognition algorithm. The simplest method is to then compare differences in the intensity of individual pixel values to each other. However, expressing an image in this way is akin to comparing two points in large-dimensional space, depending on the resolution of the image, (on the order of 10^5 dimensions or more) which is computationally inefficient and inaccurate.

Instead, we make use of eigenvectors to define subspaces or "face spaces" that can represent the images in considerably fewer dimensions. We cannot simply find the eigenvectors of this large-dimensional space, because the resulting matrix would be too large for an average personal computer to calculate (dimensions 10^5x10^5 or more). However, we can make use of Singular Value Decomposition (SVD) and Principle Component Analysis (PCA)[1] to find the singular values, relate them to the original data set's eigenvectors, and compute a much smaller matrix as a result. Then, we can make use of PCA to rearrange the eigenvectors by their relative contribution to expressing each point in the data set. This is possible because faces contain the same features and general layout, so the data points expressing many human faces will tend to be clustered together, not spread around randomly. As a result of this, some eigenvectors will point in a direction that effectively describes more data points than others. PCA allows us to identify and order these eigenvectors so that we can describe our data set with significantly less information.

Once we have performed PCA to define our subspace, we can express each image as a linear combination of the eigenvectors and weights, which allows us to simultaneously express the original images in fewer dimensions (between 10 and 100 dimensions) and emphasize the data in those images that are most relevant to differentiating one face from another. Thus, we minimize irrelevant variation and maximize relevant variation in our data set using only applications of matrices and linear algebra. This critically important concept is a major component of both the Eigenface and Bayesian Recognition algorithms.

After defining our Eigenface space with PCA, we are able to measure the Euclidean distance between images by treating each image as a point in our Eigenface space. Since we have taken steps to minimize intrapersonal differences between images, we can assume that the image that is the smallest distance from the input image belongs to the same person with a high accuracy rate (in our implementation we found this to be 90.16% accurate).

[1]An interactive visualization can be found here: http://setosa.io/ev/principal-component-analysis/

The Bayesian approach to facial recognition is similar to Eigenfaces in that we must first reduce the number of dimensions that we are considering and select only the ones that contain valuable data. The difference between the Eigenface subspace and the Bayesian subspace is that while the former captures captures the subtleties in the human face, the latter captures the important differences between two pictures of the same person. In this way, we define an "intrapersonal difference" subspace. We then use this subspace as a pre-defined condition to calculate the Bayesian normal distribution. We can then input a face image, project it onto the difference space to find the probability that the relationship between the inputted image and every other face in the data set is an intrapersonal one. Returning the image corresponding to the highest probability lead to results that were 86.89% accurate in our implementation.

In the following sections we break down the mathematical approaches of both Eigenfaces and Bayesian facial recognition. We then discuss how we applied each algorithm to a database of faces created of our classmates and professors, as well as our findings and performance analysis.

3. Face Space with Eigenfaces

Black and white digital images are made up of pixels, where each pixel can be represented as an "intensity" value between 0 and 255. In accordance with this fact, we begin by splitting a pxp image into px1 columns and appending each column to create a p^2x1 column vector. This means that an image that is 256x256 pixels can be equivalently be described as a point in 65,536-dimensional space. We then take several of these reshaped images and place them into a p^2xn matrix with one column per image. This formatting allows us to perform matrix operations, find eigenvectors, and eventually perform PCA on our data set.

Pictures of faces that are taken under similar conditions (i.e uniform lighting, head-on, and from a uniform distance) will contain similar pixel intensities. By ensuring that our original photos are as uniform as possible we have already begun to minimize unnecessary variation between images that could deceive our algorithm. The next step we take to eliminate unwanted information is to recognize that any given number of images taken in such conditions will cluster together in the large-dimensional space, so they could also be quite accurately described in a subspace of considerably fewer dimensions.

In order to describe these points in a smaller subspace, we make use of Principle Component Analysis (PCA). This technique allows us to find a small number of vectors that can describe the distribution of face image points. The eigenvectors that result from PCA create an orthanormal set that span the subspace of face images, or "face space." Each vector is a linear combination of the original face images and are also the eigenvectors of the original images' covariance matrix.

There are a few steps we must go through before we can perform PCA. Let the set of training images contained in the matrix $\mathbf{T} = [\vec{T_1}, \vec{T_2}, \vec{T_3}, ..., \vec{T_n}]$, where $\vec{T_n}$ is the nth image in the training set. We then mean-center this vector by subtracting each row by its own mean and dividing $\sqrt{p^2}$ element-wise from $\mathbf{T}$ where p^2 is the number of rows in $\mathbf{T}$. This defines a matrix $\mathbf{S}$ of mean-centered vectors. Normally at this point we would find the covariance of $\mathbf{S}$ by taking $\mathbf{S}\mathbf{S}^T$ and then solving for the eigenvectors and eigenvalues. However, the resulting covariance matrix is 65,536 by 65,536, so we are unable to calculate it using MATLAB. Instead, we can perform Singular Value Decomposition on $\mathbf{S}$ and use the information it gives to find the eigenvectors and eigenvalues.

Singular Value Decomposition (SVD) allows us to find the eigenvectors and eigenvalues of a matrix that are not square and invertible. In other words, we can use SVD to find the eigenvectors and eigenvalues without needing to create and analyze gigantic matrices. We will need these particular eigenvectors to perform PCA; it gives us most of the information we need to define a natural coordinate system to frame our facial recognition subspace. We define the SVD of a matrix $\mathbf{S}$ as

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{1}$$

where $\mathbf{\Sigma}$ is a diagonal matrix containing the squares of $\mathbf{S}$'s eigenvalues, and $\mathbf{U}$ and $\mathbf{V}$ are rectangular matrices containing the eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ respectively. We calculate the components of the SVD using the following equations:

$$\mathbf{A}\vec{v}_i = \mu_i \vec{u}_i \tag{2}$$

$$\mathbf{A}^T\vec{u}_i = \mu_i \vec{v}_i \tag{3}$$

Recall that $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ are both covariance matrices, which are by definition square, invertible, and contain information about how scattered data is about the mean. The eigenvectors of such matrices form an orthonormal basis set. So, by solving these equations for every vector in both $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$, we form the matrices $\mathbf{U}$, where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, ..., \mathbf{u}_n]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, ..., \mathbf{v}_n]$. Also, note that since the singular values provided by SVD are the singular values for both $\mathbf{U}$ and $\mathbf{V}$, we can select the eigenvector matrix corresponding to the proper covariance matrix (the $\mathbf{U}$ matrix in this case). Because $\mathbf{U}$ contains the eigenvectors of the covariance matrix $\mathbf{A}\mathbf{A}^T$, we are able to use it to calculate the PCA.

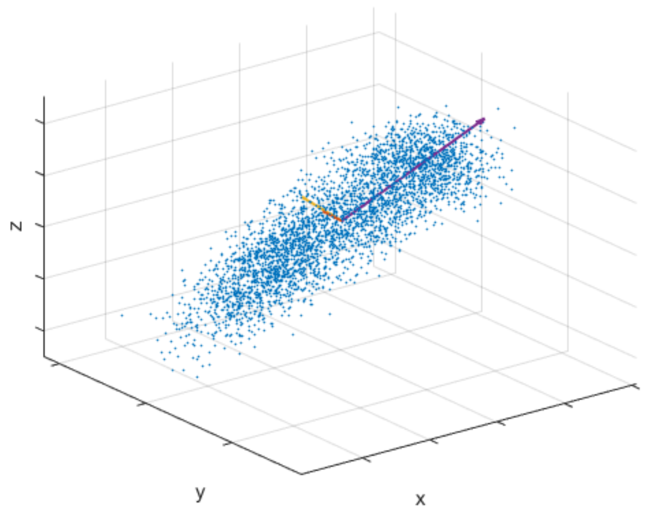

FIGURE 1. The plot shows a point cloud in a direction not of the original axis. The leading eigenvector pulled from the covariance of the data, shown in purple and overlaid on the original data, points in the direction of most variance. This can be seen with the new set of axis defined by the three orthonormal eigenvectors pulled from PCA of the original dataset.

Figure 1 is an example of the resulting vectors pulled from PCA of an arbitrary set of data. The focus is on the leading eigenvector–the large purple arrow–which points in the direction of greatest variance. The length of the vector is defined by its respective eigenvalue. Figure 1 and 2 (shown on the following page) describe what PCA is primarily useful for: data-compression. When a large portion of the data falls close to a few of the eigenvectors, some information about the data can be adequately described using only the eigenvectors that best model the data, instead of every single eigenvector. It turns out that the eigenvectors corresponding to the largest eigenvalues contain the most information about the original data.

With the eigenvalues of $\mathbf{A}$ stored in $\mathbf{\Sigma}$ and their respective eigenvectors stored in $\mathbf{U}$, and knowing that the eigenvector corresponding to the largest eigenvalue points in the direction of greatest variance in the data, reorganizing $\mathbf{U}$ in order of descending eigenvalues and selecting the first k columns to describes a k-dimensional subspace. If k is chosen correctly, the subspace will contain only the most important information about the data set. This

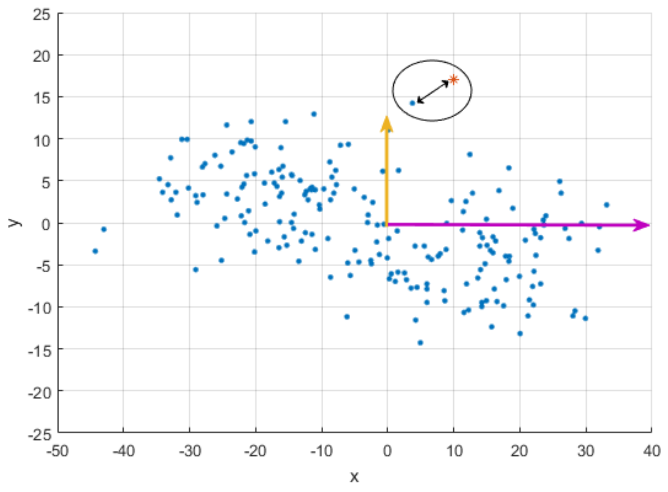

FIGURE 2. Suppose the data from Figure 1 is a 3D representation of a face set. Then Figure 2 shows a data compression, also known as a 2D data projection, of the first two eigenvectors pulled from the PCA of said dataset. The red star is an example of a face projected onto that space and the arrow is indicative of the Euclidean distance between the two. Because the points are closest to each other, we can imply that they both belong to the same face.

allows us to dispose of our original N^2-dimensional space, which contained a lot of superfluous data and provided no information to the algorithm about what data was important.

Once we have our face space, the following set of operations should be performed on both the training data set and the test data set. This is so that both are expressed in the same subspace and we cannot compare a vector expressed in an extremely large-dimensional space to a similar vector expressed in a much smaller-dimensional space.

A new face image $\vec{T}$ (a column vector with N^2 components) is projected onto the face space by the k number of eigenvectors calculated during PCA. Before we project $\vec{T}$ onto the face space we must first mean-center it using $\vec{\Psi}$, the mean of training set of faces used for the PCA calculation. The computation can be expressed by the equation

$$\vec{\Omega} = \mathbf{U}^T(\vec{T} - \vec{\Psi}) \tag{4}$$

where $\mathbf{U}^T$ is the matrix of eigenvectors and T is the face image. This 'projection', also known as the linear combination, outputs $\vec{\Omega}$, a vector of dimensions kx1 that describes the contribution of each eigenvector in describing the new face in the face space.

Given the matrices that represent the training and test data sets in the same subspace, we can test the for the Euclidean distance between any two points, or faces, in these subspaces and find the point that lies closest to the face that corresponds to our test image. Now we can give our Eigenface algorithm a single image, and it will search a dataset and (usually) return another picture of the same person. We use the following equation,

$$e = \left(\sum_{i=1}^{k} (\Omega_{Dk} - \Omega_{Tk})^2 \right)^{1/2} \tag{5}$$

where e is the Euclidean distance between the given image Ω_T and a selected image from the database Ω_D. Performing this calculation sequentially through each database image results in a row vector whose entries are the distance between each image in the database and the test image. The column with the smallest value of e returns the index of the image that is most alike the given image under the premise that similar pictures of the same individual will have fewer differences than a pair of pictures of different individuals.

4. Bayesian Facial Recognition

The key idea of Bayesian facial recognition is to compare two images based on the probability that they either belong to the same person or to different people based on pre-existing conditions that we set. Typically Bayesian facial recognition algorithms calculate both the probability that two images belong to the same person as well as the probability that they belong to different people. This can be useful for facial recognition algorithms that do not contain complete datasets. For example, if the algorithm is presented with a face that does not correspond to a face that is already in the database, the Bayesian algorithm will know to append that face to its database instead of search for the next closest match. However, since our database is complete there is no need to add new faces to it, so we only need to calculate the intrapersonal probability, or the probability that two faces belong to the same person.

We begin again with the same process as Eigenfaces, by first reshaping our 3D array into a 2D matrix of images represented as points in high-dimensional space with dimensions p^2xn, where n is the number of pixels along the x, and y axis and n is the number of images in the training set.

Each set of images in the training set is a slightly different picture of the same individual. So if we had 30 individuals we would have a set of 60 images to represent the base training set.

Let **D** be a matrix that holds the difference of the two images contained in the original dataset. This operation reduces the size of the dataset to be $p^2 \text{x} \frac{n}{2}$. We again mean-center the matrix by subtracting the mean of each component from itself and dividing by $\sqrt{p^2}$. We perform PCA on the matrix to define our face space in order to do away with the less important variation contained in the dataset as we did in the Eigenface approach. From the analysis we will again pull k number of orthonormal eigenvectors to define a face space. This face space will serve as the point of reference for the algorithm; it tells the computer what should be considered an intrapersonal relationship (an image of the same person).

We then project all of the faces in our data set and test set onto this low-dimensional face space by using Equation 4.

For implementation we'll be using Bayes' Law,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6}$$

which can be naively simplified as the probability of A given new evidence B, but due to the fact that $P(B)$ can be computationally challenging, and in this case moot, all we care about is the numerator which gives us the probability that Z is of the ω_i class. Equation [7][2] can be interpreted as the probability that z_j, of set Z, is of the same class as m_{ij} given that ω_i is true.

$$p(Z|\omega_i) = \frac{1}{(2\pi)^{m/2}\prod_{j=1}^{m}\sigma_j} exp\Big(-\frac{1}{2}\sum_{j=1}^{m}\frac{(z_j - m_{ij})^2}{\sigma_j^2}\Big) \tag{7}$$

The few important things to note between the two equations would be ω and Ω which represent the assumption being made, defined as the intrapersonal relationship, in order to draw the conclusion that $z_j \in \omega$,[3], or $\vec{I}_T \in \Omega$ which implies that the test image is intrapersonally related to the data set image. In Equation 7, σ_j are elements of the covariance, similar to what we see in the vectorized version[4] of the same equation in 8.

Now, instead of calculating the Euclidean distance between the test image and each dataset point, we use the training subspace and the difference

[2]This equation was taken from reference [4].

[3]Which reads as x_j is an element of ω

[4]This equation appears in reference [1,4].

$\Delta = \vec{I}_T - \vec{I}_D$, where $\vec{I}_T$ and $\vec{I}_D$ are the compressed test image and a compressed data set image respectively. We then use Bayes' Law to calculate the probability that $\vec{I}_D$ and $\vec{I}_T$ belong to the same person. We can estimate the probability to be represented by the Bayesian normal distribution in equation 8

$$p(\Delta|\Omega) = \frac{1}{(2\pi)^{k/2}|\Sigma_\Omega|^{1/2}}\exp\Big(-\frac{1}{2}(\Delta-\mu_\Omega)'\Sigma_\Omega^{-1}(\Delta-\mu_\Omega)\Big) \tag{8}$$

where Σ_Ω is the covariance matrix of compressed intrapersonal face data transposed, μ_Ω is the mean of the difference faces projected onto the intrapersonal face space and k is the number of dimensions in the subspace. This can be interpreted as

We use this equation to compare the given test image to every data set image. The result of this operation is a row vector containing the resulting probabilities. Selecting the column that corresponds to the largest probability will return the matching face.

5. Implementing Eigenfaces

To begin we were given a 3D array of dimensions 256x256x132, where every two images were of the same person, signifying 66 people. We extracted two sets of data, each containing one photo of each person, and reshaped each into a 65,536x66 matrix. Mean-centering and normalizing the training data set creates the matrix **A**, where **A** has dimensions 65,536x66. We then defined the covariance matrix C to be $\mathbf{A}\mathbf{A}^T$ because we need to find eigenvectors that describe the relationship between pixels as opposed to the relationship between images.

We then perform PCA on **C** and selected the first 37 ordered eigenvectors, resulting in a 65,536x37 matrix of mean-centered, orthanormal eigenvectors describing our subspace. We chose to use 38 dimensions by running our algorithm for different values of k and picked the value with the lowest run time and the highest possible accuracy rate. Running the model by cutting an increasing amount of leading eigenvectors showed to have an effect on the accuracy by cutting away less-useful variation in the data. A second analysis suggested that using eigenvectors 2 through 37 would lead to better accuracy.

We then project both the training and test data sets onto our 35-dimensional subspace by plugging into equation 1. This creates two 35 by 66-dimensional matrices, where each column contains the weights to express the image in the 35-dimensional subspace. We can then select an image from the compressed test database at random and compare it to every image in the training set,

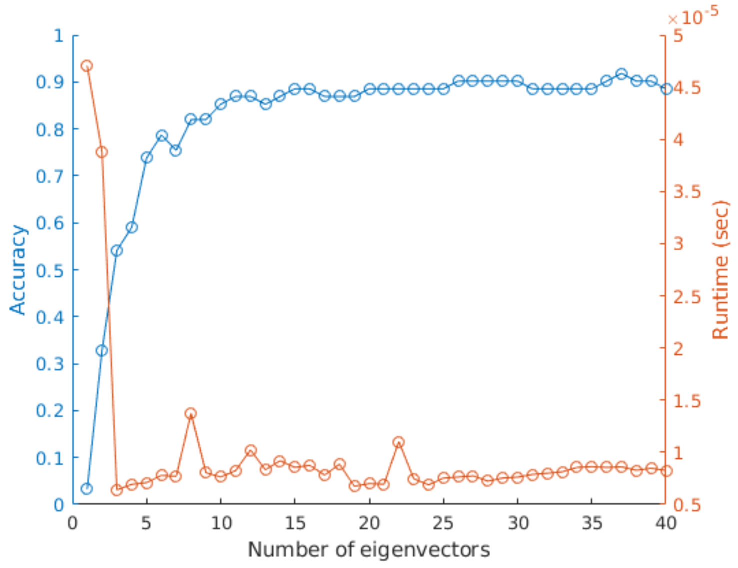

Figure 3. The figure shows the accuracy and average runtime of using k number of eigenvectors to define the face space with. Diminishing returns on performance begins around 37 eigenvectors

calculate the Euclidean distance and store it as an entry in a row vector of dimensions 1x66.

5.1. **Performance Analysis.**

We found that compressing the data to a 35-dimensional subspace resulted in the entire algorithm being able to identify a match between one test face and the database set with a 90.16% accuracy rate and a run time of approximately 0.000001 seconds to identify one face.

Figure 3 presents the results of running the current Eigenface method of facial recognition using MATLAB. Accuracy is determined as the number of correct matches divided by the number of total faces in the test set. Accuracy could theoretically be increased by expanding the training set to include more faces of people in the test set so as to give more points to match to with a new given test face.

The Eigenface algorithm is fast and efficient, and generally works well under controlled conditions. However, since it only accounts for Euclidean distance between two points, its method of determining when two faces belong to the same person is based on the assumption that two images of a person will appear almost identical, without regard for factors like inconsistent lighting, different angles and backgrounds, and the presence or absence of facial hair or accessories. Choosing the right eigenvectors from the PCA helps eliminate unnecessary variation in the dataset. Also, if the algorithm is presented with a face that belongs to someone who has not been added to the database, the Eigenface algorithm is only capable of returning the next closest match, while the Bayesian facial recognition algorithm could decide to append the face to its database.

6. Implementing Bayesian Facial Recognition

We started with a 3D array of images of dimensions 256x256x222. Each member of the class had at least two photos of themselves in this data set, though some had up to 4. We selected a single pair of images for every person in the data set, subtracted pixel intensities from one image from the other, and added the resulting differences to a matrix of training "difference images" of size N^2x43.

We then normalized this matrix and found the covariance, performed PCA, and selected the eigenvectors corresponding to the first 32 largest eigenvalues. We chose 32 eigenvectors to define our "intrapersonal difference space."

We then split our test data set into two matrices, so that each person in the class had one of themselves in each set. After normalizing them, each set is then passed through equation 4, where $\mathbf{U}$ is the matrix of 65,536x32 matrix of eigenvectors and the resulting Ω is the compressed 32x1 image. By using equation 8, we were then able to select a random image from one set and search through the other set of images to find the highest probability that the image belonged to the same person.

6.1. **Performance Analysis.**

Figure 4 shows the results of our model where we defined Δ, the intrapersonal relationship, using a smaller subset of faces than what was in the entire dataset. Note that there is a point where increasing the number k eigenvectors evidently increases computationally inefficiency and also decreases accuracy.

We found that using 1 through 34 eigenvectors to define the face space proved to be the most fruitful for this algorithm in the first plot. In an attempt to eliminate more unnecessary variation, such as variations in hair, skintone, and lighting, we plotted the accuracy of using fewer leading eigenvectors. The results suggests that using eigenvectors 4 through 34 would yield the most accuracy. With a single recognition runtime of 0.00007 seconds and an 86.89% overall accuracy it shows to be less accurate and slower than basic Eigenfaces.

Implementing Bayesian statistics for facial recognition proves to be a less effective and accurate way of comparing faces in these two alike datasets.

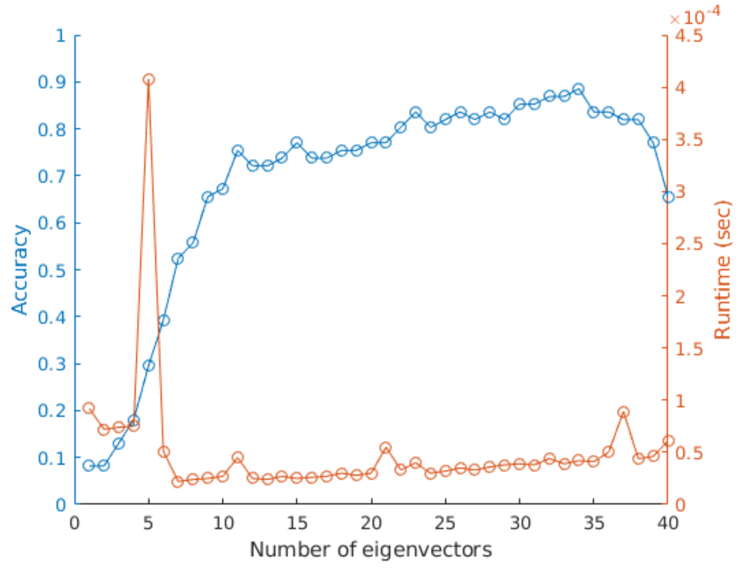

FIGURE 4. The figure shows the accuracy and average runtime of using k number of eigenvectors to define the face space with. Diminishing returns on performance begins around 35 eigenvectors

One reliable aspect of this method is that the result is given in a probability instead of a computational 'guess' as to which face belongs to who.

7. Conclusion

While the Bayesian implementation proved to be a more practical approach for facial recognition, in the sense that it has more application than typical Eigenfaces, more tests would need to be made to fully assess the key use case of the algorithm in the real world. The datasets given to test the accuracy of each approach were all fairly catered to for use in computation and comparison and probably had a non-negligible amount of influence on the results. The next step for the Bayesian approach would be to implement Fisher's linear discriminant analysis after conduction PCA which better emphasizes the *differences* between classes historically yielding better results.

7.1. **References.**

Balas, Benjamin. "Bayesian Face Recognition and Perceptual Narrowing in Face-Space." Developmental Science 15.4 (2012): 579–588. PMC. Web. 2 Apr. 2018.

[2] B. Moghaddam, C. Nastar, A. Pentland, "A Bayesian Similarity Measure for Deformable Image Matching", Image and Vision Computing, Vol. 19, Issue 5, May 2001, pp. 235-244 Web. 2 Apr. 2018

[3] M. Turk, A. Pentland. "Eigenfaces for Recognition", Vision and Modeling Group, The Media Laboratory, MIT 1991. Journal of Cognitive Neueroscience Volume 3, Number 1. 2 Apr. 2018

[4] C. Liu and H. Wechsler, "A Unified Bayesian Framework for Face Recognition", Department of Computer Science, George Mason University, 0-8186-8821-1/98 0 1998 IEEE pg151-155.