Finding the Optimal Training Set for Eigenfaces

Nathaniel Yee, Eric Miller

I. Abstract

We analyze the effectiveness of various training and testing datasets for the Eigenfaces algorithm. Contrary to the belief that more data leads to better results, we examine a case where Eigenfaces performs comparable with less data. This occurs when the features of the testing set can be found in similar photos in the training set. On the other side of the spectrum, if the testing set has features not contained in any one other emotion, we show that more data leads to much better results. Overall, training on more data leads to a more accurate, larger feature capturing, facial recognition program.

II. Examining eigenfaces for different training/testing datasets

In this paper, we will measure the effectiveness of Eigenfaces' facial recognition when trained and tested on various datasets (see table 1). Since principal component analysis (PCA) attempts to find the maximum variation between all training images, if we include multiple images of the same person, PCA will attempt to find variation between images of the same person. We propose that for a given test set, we want a training set consisting of very similar images. If no training images are similar in features, we want to use a training set consisting of many varied expressions to try and capture a wider variety of features.

Table 1: Training and Testing Datasets

Test index	Training images	Testing images
A	Neutral All except Happy	Happy
B	Neutral All except Disgusted	Disgusted

Table 1 shows the various training and testing datasets we use in various tests. Test index refers to the specific test. Training images refers to what images we will use to train our eigenfaces. In this paper, each test will contain two different training sets. Testing images refers to the images we will use to test the eigenfaces. In a specific test, each training set will be tested against a specific testing set.

In test A, the "happy" test images contain features similar to those of the neutral faces. We expect that training on just the neutral faces will find the variance between each person, rather than withing each person's images. So, this test should show that training on less data can lead to a better facial recognition program.

In test B, the "disgusted" test images are very different than the neutral faces. We expect that training on the neutral faces will not be able to capture the features within the disgusted faces. However, if we train on many expressions of each person, we will be able to capture these "disgusted" features. This test should show that training on more data can lead to a better facial recognition program.

Fig. 1: Three different emotions for images of Chris. Neutral, happy, and disgusted. Note that neutral and happy contain similar features while neutral and disgusted do not share many features.

III. What are eigenfaces?

EIGENFACES (algorithm) was originally developed in 1991 by Matthew A. Turk and Alex P. Pentland. It is recognized as the original, effective, near-real-time face tracking and recognition algorithm[1]. In the context of this paper, we will examine the recognition component. In particular, we will look at how the size/type of training set affects the accuracy of recognition.

Within the algorithm, the **eigenfaces** are a set of orthonormal vectors in N-dimensional (where $N = m * n = 256 * 360 = 92,160$) space. This set of eigenfaces correspond to the principal components of the pixel-pixel covariance matrix constructed from the image training data. As a result, the eigenfaces capture the direction and scale of maximum variance throughout the training data. Their corresponding **eigenvalues** represent the amount the data varies in the direction of that eigenface.

We consider a **compressed image** to be the k most significant components of the projection of an image vector into the coordinate frame formed by the eigenfaces. Compressing an image is done by finding the projection of the image onto the first **k** eigenfaces, and decompressing it requires summing the eigenfaces with coefficients determined by the elements of the compressed form.

To perform facial recognition, we calculate the closest compressed training image (by Euclidean distance) to the compressed test image.

IV. The eigenfaces algorithm

A. Overview

THE Eigenfaces algorithm describes a method of using PCA to calculate the distance between two images by first projecting them into the k-dimensional eigenspace. This compressive process significantly reduces the amount of data that must be stored and parsed for each training example allowing real time facial recognition. Second, it massively improves accuracy by reducing the impact of lighting conditions, background colors, and expressions because the principle components store critical axes of variation between the training data.

To implement Eigenfaces, one must follow a three-step process. First use PCA to determine the optimal eigenfaces in a set of **training data**. Second, use the eigenfaces to compress and store a set of labelled **matching data**. Third and finally, perform facial recognition by projecting a new image onto the same k-dimensional eigenspace and finding the closest person by Euclidean distance in the **matching data**.

In most implementations of Eigenfaces, the **training data** and **matching data** are taken to be identical, but this does not need to be the case, allowing for easy addition or removal of faces from the **matching data**. This is possible because the eigenfaces converge to represent the generic variability between any two human faces, and their nature does not depend strongly upon the specific faces used as training data.

B. Finding eigenfaces

The process for finding eigenfaces from the training data is based on performing a Principle Component Analysis (PCA) of the pixel-pixel correlation of the **training data**. As with any PCA, this means we seek the first k eigenvectors (and corresponding eigenvalues) of the correlation matrix

$$RR^T$$

where R is a matrix containing as columns the flattened vector representations of the training images.

Unfortunately, because R has a N rows ($92,160$ in our case), RR^T has dimensions $N \times N$, and representing the correlation matrix naively in memory would require many tens of gigabytes of RAM. Additionally, because finding eigenvectors is an $O(n^3)$ operation, calculating the eigenvectors directly would likely take many days.

Fortunately, calculating the eigenvectors of $R^T R$ is much easier and, as we will now show, deriving the eigenvectors of RR^T from those of $R^T R$ is trivial.

Consider some eigenvalue λ and eigenvector x of $R^T R$ such that

$$(R^T R) \times x = \lambda x$$

Left-multiplying by R and regrouping yields

$$(RR^T) \times (Rx) = (R\lambda) \times x$$

Because matrix-scalar multiplication is commutative,

$$(RR^T) \times (Rx) = \lambda(Rx)$$

which shows that Rx is an eigenvector with eigenvalue λ of RR^T.

This insight allows us to calculate the first k eigenfaces by first computing the the eigenvectors of the image-image correlation matrix, then multiplying each by R, a much more efficient process than trying to compute the principle components directly.

C. Compressing matching data

Now that we have the eigenfaces, we can find the compressed form of each image in the **matching data** by taking the projection of the matching image onto each of the eigenvalues, and packaging the results together into a column vector, giving the compressed form of the image. These are stored with labels denoting the identity of the person each image represents.

In this paper, in order to compare the goodness of the eigenfaces on level ground, we use one image per person (usually the first) as our matching data.

D. Recognizing new images

When given a new image I to recognize, Eigenfaces first compresses I in the same way the **matching data** was compressed in **??**. Then, it finds the nearest neighbor of the compressed test image in the compressed **matching data**, and returns the label of that image.

E. Limitations

Fig. 2: The image representation of an eigenface trained using the neutral faces.

V. Figures of merit

In order to understand the performance of the various incarnations of Eigenfaces, several quantitative measures are useful.

Fig. 3: An image of Eric that has been expressed as linear combinations of the 30 most significant eigenfaces. Since our training set contains people wearing glasses, we see glasses frames in the projected image of Eric.

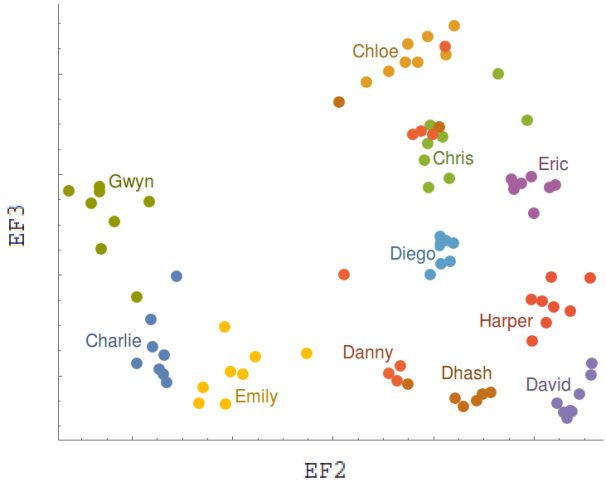

Fig. 4: Shows a graphical representation of people projected onto the second and third eigenface. Each point represents an image of a person. Each color represents a person. Notice that each person's images cluster together. If we were to perform facial recognition using these eigenvectors, we would get decent results due to the small euclidean distances between images of the same person

A. Accuracy

The first and most obvious measure of algorithmic efficacy is the accuracy algorithm, defined as

$$Acc = \frac{\text{\# of successful recognitions}}{\text{\# of attempted recognitions}}$$

where success is defined as the algorithm's closest chosen closest match being an image of the same person as the given test image. Because both the test and training data are extracted from the same raw dataset, labelling that dataset at the beginning and maintaining those labels throughout the computation makes this analysis trivial.

Unfortunately, because Acc is always an integer multiple of $1/|test\ data|$, it often does not provide enough granularity to distinguish small changes in algorithmic effectiveness.

B. Recognition confidence

To create a more finely-discerning version of Acc, we can analyze the confidence of the algorithm in selecting the match it did for compressed test image t. This involves looking at two compressed images from the training set: c, the nearest training image of the correct person, and w, the nearest training image of *not* a correct person. Because Eigenfaces chooses the closest training face in k-space, if the distance from the test point to c is less than the distance to w ($|t-w| > |t-c|$), then the algorithm will return the correct match. From this insight, we can compute the Recognition Confidence

$$RC = \frac{1}{|T|}\sum_{t\in T} |t-w| - |t-c|$$

where T is a set of test images.

Notice that each term of this average is positive if eigenfaces would return a correct match for that test image, and negative if it would fail. Thus, larger values of RC correspond to the algorithm being more likely to choose the correct match.

This provides better granularity than the raw Acc measurement, but loses the direct connection to the performance of the algorithm.

C. Clustering coefficient

At its core, Eigenfaces is an algorithm for projecting the images onto k-dimensional compressed faces such that the compressed faces of the same person are clustered together. To directly analyze the effectiveness of this clustering, we can compute for each person in the dataset the shortest tour distance between the k-dimensional vectors corresponding to each of their images, then average those distances to create a **clustering coefficient** (CC). Intuitively, this represents the size of each cluster, so smaller values are better. Importantly, values of the clustering coefficient are only comparable if the same number of images per person are used, so for all values of the clustering coefficient used in this paper, it was computed from the full 8 faces per person.

VI. Performance of two training sets

Now test Eigenfaces for tests A and B (see table 1).

A. Accuracy

As seen in fig 5, having multiple images of each person does not improve accuracy. In this case, both training sets lead to near 100% accuracy when we train on the similarly looking neutral faces.

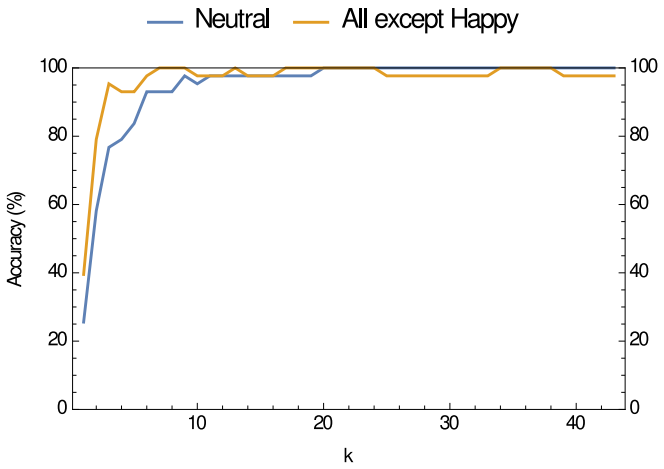

Fig. 5: Shows the results of test A. In this case, both algorithms perform comparably.

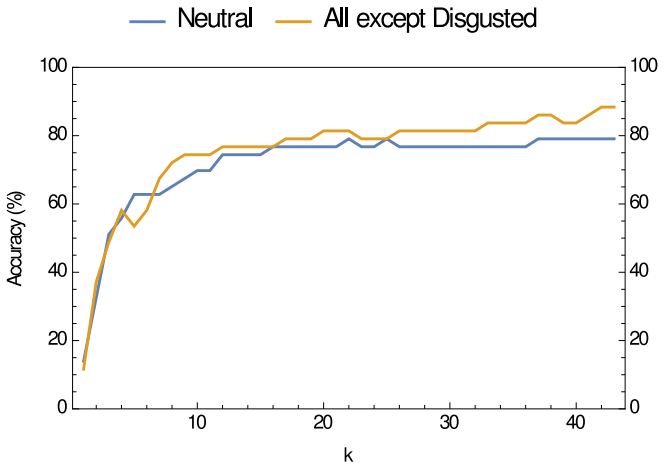

Fig. 6: Shows the results of test B. In this case, training on more expressions leads to better results as k gets larger.

However, as seen in fig 6, when we our testing set is a much more varied expression, it is optimal to train on many faces and use a higher k value.

B. Recognition confidence

C. Clustering coefficient

Next we will compare the clustering coefficients.

Table 2: Clustering Coefficients

Number images	Training images emotions	Clustering coefficient
41	Neutral	173
301	All except happy	201
301	All except disgusted	200

As shown in table 2, if we increase the number of images and emotions, the clustering coefficient increases from 173 to 200 or 201. This confirms part of our initial assumption; as we increase the number of images per person, eigenfaces captures more variation between images of the same person and ultimately creates larger clusters. If we want to test on one type of emotion, happy, we should train on a similar expressions, neutral. However, if we want to capture more raw features, we should train on more images of more emotions.

VII. Overarching Conclusions / Insights

Throughout the paper, we have observed the effectiveness of Eigenfaces for various training and testing datasets. In terms of pure accuracy, Eigenfaces can get away with a small training set if it is similar to the training set. For example in test A, we achieve close to 100% accuracy for both training sets. If the testing set has outlandish features, it better to train on a larger, more varied, training set. For example in test B, the larger training set is about 10% better than the smaller training set.

VIII. Appendix: 3D cluster visualizations

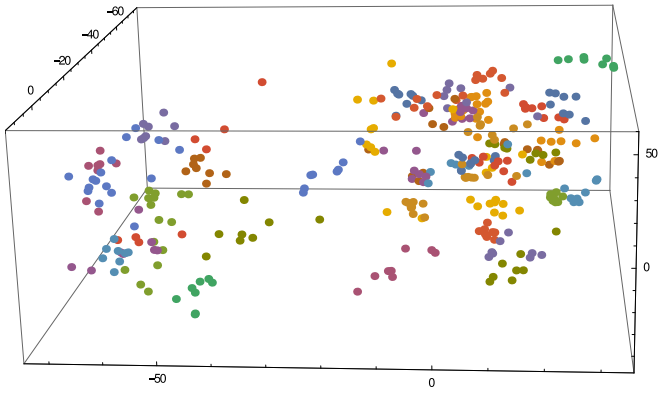

Fig. 7: 3D plot of images projections onto second through fourth eigenfaces

IX. Appendix: Choosing k (number of eigenfaces)

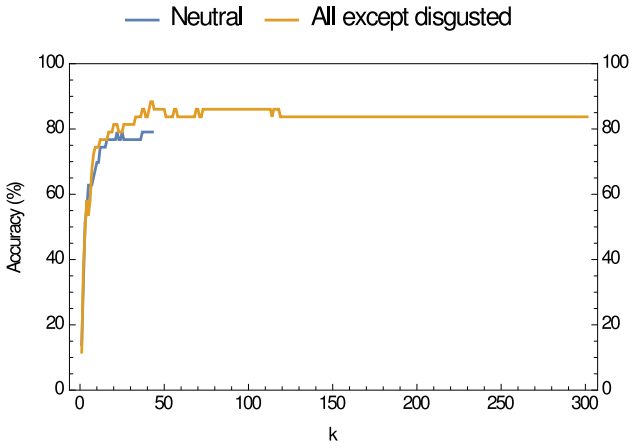

Fig. 8: Test B for all possible values of k