NOTES ON PCA EIGENFACE

1. The aim of Principle Component Analysis (PCA) is to reduce the dimensionality of the data while retaining as much information as possible in the original dataset.
2. PCA is a mathematical procedure that uses an orthogonal transformation to convert a set of `N` face images into a set of `K` uncorrelated variables called Eigenfaces.
3. This transformation is defined in such a way that the first Eigenface shows the most dominant "direction" or "features" of the training set of images. Each succeeding Eigenface in turn shows the next most possible dominant "direction", under the constraint that it be uncorrelated to the preceding Eigenface.
4. To reduce the calculations needed for finding these Eigenfaces, the dimensionality of the original training set is reduced before Eigenfaces are calculated.
5. Since Eigenfaces show the "directions" in the training set, and each proceeding Eigenface shows less "directions" and more "noise", only the few first Eigenfaces, `K`, are selected whereas the rest of the last Eigenfaces are discarded. Hence, `K < N`.
6. The goal is to represent a face image as a *linear combination* of a set of eigenvectors. These Eigenfaces (eigenvectors) are the PCA of the training set of face images generated after reducing the dimensionality of the training set.
7. Once Eigenfaces are selected, each training set image is represented in terms of these Eigenfaces. When an unknown face comes for recognition, it is also represented in terms of the selected Eigenfaces.
8. The Eigenface representation of an unknown face is compared with that of each training set face image. The "distance" between them is calculated. If the value is above a specified threshold, the unknown face will be recognised as a person in the training set.

DISADVANTAGES

> *"As a sneak preview, it seems that head-tilting, eye-closing, and smiling account for a lot of variability. If the reader is a fugitive on the run (and somehow find's time to read this blog; I'm so*

honoured!), and wishes to have his photograph not recognised, he should smile, tilt his head, and blink vigorously while the photograph is being taken." – j2kun [1]

"The eigenface method has problems identifying faces in different light levels and pose positions. Better results are obtained when the input faces are preprocessed to remove background details, and all faces are equally oriented geometrically.The face must be presented to the system as a frontal view in order for the system to work." – Sajan [5]

SUMMARY OF EIGENFACES ALGORITHM

1. The face images must be centred and of the same size. Flatten `N` training set images into vectors placed as the rows of a face matrix. Each row in the `N × M²` matrix represents the pixels of a training image.
2. Calculate the mean face vector, `Ψ`, then subtract it from each row of the face matrix to get the corresponding normalised face matrix, `A`.
3. To calculate the eigenvectors, we need to calculate the covariance matrix, `C = Aᵀ·A`, where A is the normalised face matrix of dimension `N × M²`. The dimension of `C` is `M² × M²`, which is very large and will require a lot of computation time.
4. To reduce the calculations and the effect of noise on the needed eigenvectors, we can compute them from a covariance matrix of *reduced dimensionality*, `C = A·Aᵀ`, which results in a dimension of `N × N`.
5. Step 3 is only preferable over step 4 when we have more images than pixels. Though in reality, we will often have more pixels than images.
6. Perform Singular Value Decomposition (SVD) on `C` to obtain the eigenvectors and eigenvalues.[1] If `C` is a covariance matrix of *reduced dimensionality*, we need to perform an extra step to get the desired eigenvectors, `U`. Compute `U = Aᵀ·V` where `V` is the "temporary" matrix of eigenvectors obtained from performing SVD on `C` of *reduced dimensionality*. The Eigenfaces matrix `U` is of `M² × N` dimension.
7. Since we are looking for the top `K` eigenvectors, we will need to sort the eigenvectors by eigenvalues in descending order.
8. Select `K` best eigenvectors/Eigenfaces (those with largest associated eigenvalues) empirically, such that `K < N` and can represent the whole training set. One way to do this is to plot each eigenvalue against its index number and note the index where the change in eigenvalue start to diminish.[2] `U` is now a `M²`

$\times K$ dimension matrix.

9. Calculate the set of weights, W, associated with each training image. This is done by computing the dot product between the normalised face matrix A and the Eigenfaces, $W = A \cdot U$, producing a $N \times K$ weights matrix.
10. We can now reconstruct any face since it can be expressed as a *linear combination* or weighted combination of all the Eigenfaces in addition to the mean face vector Ψ, as long as we have enough training faces and Eigenfaces.
11. To reconstruct a face, compute the dot product between the face's weights (a row in W) with the transpose of Eigenfaces and add back the mean face. $W_i \cdot U^T + \Psi$ will produce a matrix of $1 \times M^2$ dimension containing the pixels of the reconstructed image.
12. To recognise a new face, convert the input image into a face vector, y, and normalise it (subtract mean face vector from it). Project the normalised face vector onto the Eigenfaces matrix U to obtain the weights vector. This means computing $w = y \cdot U^T$ where w is of $1 \times K$ dimension.
13. Calculate the average sum of squared differences between the new weight vector w and the weight vectors of the training images in matrix W.[3] The minimum average value will indicate a matching face given that is is within a set threshold.

REFERENCES

Some helpful references in descending importance:

1. Eigenfaces, for Facial Recognition https://jeremykun.com/2011/07/27/eigenfaces
2. Eigenfaces and Forms https://wellecks.wordpress.com/tag/eigenfaces
3. How PCA Recognises Faces – Algorithm in Simple Steps https://www.youtube.com/watch?v=SaEmG4wcFfg
4. Eigenfaces for Dummies http://jmcspot.com/Eigenface
5. Face Recognition in Python http://pyfaces.blogspot.my
6. Eigenfaces https://en.wikipedia.org/wiki/Eigenface
7. Eigenface-based facial recognition http://openbio.sourceforge.net/resources/eigenfaces/eigenfaces-html/facesOptions.html
8. Face Recognition Using Eigenfaces http://courses.cs.washington.edu/courses/cse576/08sp/projects/project3/artifact/ankit/artifact/index.html

1. `np.linalg.eigh(mat)` function returns a set of eigenvalues and eigenvectors.
2. The number of Eigenfaces to be used can be decided by looking at the eigenvalues. The eigenvectors corresponding to small eigenvalues only contain information about detailed differences in comparison to a higher level discrimination by the eigenvectors with larger eigenvalues. We can choose to remove the vectors with smaller eigenvalues without affecting the results by a great amount.
3. The Euclidean distance between two weight vectors $d(w_i,w_j)$ provides a measure of similarity between the corresponding images `i` and `j`. Mahalanobis metric can also be used as an alternative to the Euclidean metric.