Principal Component Analysis (PCA): Face Recognition

Austin Zhong

November 2022

Abstract

Principal component analysis (PCA) is one of many tools in the machine learning toolbox. It is used to reduce the dimension of complex data sets using linear algebra, to identify underlying trends. Most notably, PCA projects data points onto pricipal components using eigenvalues from the covariance matrix of the data. In this paper, we will take a mathematical approach towards PCA as well as explore one of it's real world applications in facial recognition called Eigenfaces.

1 Introduction

Principal component analysis is an unsupervised machine learning method used in a variety of subjects. It is typically used for analyzing large datasets that have high dimension/features per observation, enabling the visulaization of high dimensional data. Imagine working with a data set with hundreds of entries, each with several observations. Natural questions would be: Is this data related to each other in any way? How can we visualize this? By shrinking down the dimension to 2 principal components, data points can be graphed on a new coordinate system, and clusters can signify trends. Principal component analysis has applications in many fields such as population genetics, microbiome studies, and atmospheric science [4].

One application of PCA is called Eigenfaces, which was developed by Sirovich and Kirby in 1987 and used by Matthew Turk and Alex Pentland in face classification in 1991 [5][7]. Eigenfaces is an interpretable application and was one of the early ideas in tackling computer vision facial recognition/detection problems. Eigenfaces determines the variances of faces in a training data set of faces, and uses the variances to encode and decode faces. At the same time, the method is utilizing PCA's strength in reducing the complexity of the data set, and reducing the number of computations.

In order to preserve the most information in the data, a covariance matrix is constructed. The covariance measures the relationship between two variables and is unbounded. Using eigenvalue decomposition on the covariance matrix grants an orthogonal basis. Typically, the greatest two eigenvalues and corresponding eigenvectors are used to transform the original data set. The result is a set of coordinates able to be graphed on principal component 1 (PC-1) and principal component 2 (PC-2) axes [1].

In section 2, we will review some linear algebra and basic statistical concepts as well as some essential theorems. In section 3, we will review the general algorithm of PCA, focusing on maximizing the variance through the largest eigenvector. In section 4, we will apply PCA on a simple dataset. In section 5, we will highlight the Eigenface method using PCA.

2 Background

In this section we will build the mathematical framework behind PCA, including vector projections and the covariance matrix, as well as the Lagrange Multiplier Theorem and Spectral Theorem.

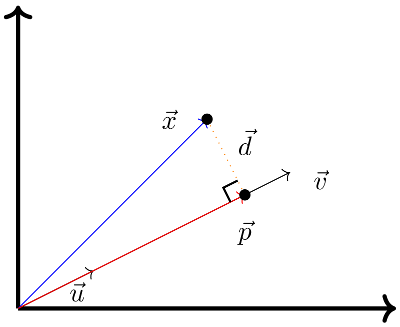

Figure 1: Projecting $\vec{x}$ onto $\vec{v}$

Definition 2.1. The *vector projection* of a vector x onto a nonzero vector v is an orthogonal projection onto the span of the unit vector in the same direction as v. Two vectors are *orthogonal* if their inner product is zero.

Lemma 2.2. *The projection vector $\vec{p}$ is given by*

$$\vec{p} = \frac{(\vec{x} \cdot \vec{v})}{|\vec{v}|}\vec{u}$$

where $\vec{x}$ is projected onto $\vec{v}$ and $\vec{u}$ is the unit vector of $\vec{v}$.

Proof.

$$\begin{aligned} p &= ku && (k \text{ is some constant}) \\ p + d &= x \\ d &= x - p \\ &= x - (ku) \end{aligned}$$

From vector multiplication,

$$\begin{aligned} p \cdot d &= 0 && (\text{since p and d are orthogonal}) \\ (ku) \cdot (x - ku) &= 0 \\ k \cdot x \cdot u - k^2 u \cdot u &= 0 \\ x \cdot u - ku \cdot u &= 0 \\ x \cdot u - k &= 0 \\ x \cdot u &= k \end{aligned}$$

$$\vec{p} = k \cdot u = (x \cdot u)u = \frac{(\vec{x} \cdot \vec{v})}{|\vec{v}|}\vec{u}$$

□

In PCA, mapping samples onto a single vector greatly reduces the dimensionality, to a direction and magnitude for each sample.

Next is the Lagrange Multiplier Theorem, essential for choosing the best principal components that preserve the most variance in the data.

Theorem 2.3 (Lagrange Multiplier Theorem)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ be the objective function, $g : \mathbb{R} \to \mathbb{R}^c$ be the constraints function, both having continuous first derivatives. Let x^* be an optimal solution the the following optimization problem such that $rank(Dg(x^*)) = c < n$ ($Dg(x^*)$ denotes the matrix of partial derivatives, $[\partial g_j/\partial x_k]$ [2].*

Single Constraint Lagrangian Function: $\mathcal{L}(x, \lambda) = f(x) - \lambda g(x)$

$$\begin{aligned} \mathcal{L} &= \textit{Lagrangian} \\ \lambda &= \textit{Lagrange Multiplier} \\ g(x) &= \textit{equality constraint} \\ f(x) &= \textit{function} \\ x &= \textit{integer} \end{aligned}$$

By taking the partial derivative of the right side of the equation, we are able to maximize the objective function with the added equality constraint.

Definition 2.4. The *derivative of a matrix* is defined as:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

(Important PCA takeaway: $\dfrac{d}{dx} x^T Ax = 2Ax$)

Definition 2.5. The *covariance* measures the direct relationship between two variables. A positive covariance signifies a positive relationship while a negative one signifies a negative relationship. A covariance equal or close to 0 means there is little to no correlation between the two variables.
Covariance is given as follows, for two variables X and Y:

$$Cov(X, Y) = \frac{1}{n-1} \sum_{i-n}^{n} (X_i - \overline{X})(Y_i - \overline{Y})$$

The *variance* is a special case of the covariance. It quantifies the spread in the data. Given set S, this can be calculated by

$$Cov(S, S) = Var(S) = \overline{S^2} = \frac{1}{n-1} \sum_{i=1}^{n} (S_i - \overline{S})^2$$

Definition 2.6. The *covariance matrix S* is a symmetric matrix giving the covariance between each pair of elements of a given random vector. It's closed form is as follows:

$$S = \frac{1}{N} \sum_{i=1}^{N} (X_i - \overline{X})(X_i - \overline{X})^T$$

N corresponds to the number of samples in $\mathbb{R}^D$, while X_i - i^{th} vector of the sample set, with dim $D \times 1$. The resulting covariance matrix S will be of dimension $D \times D$.

Example 2.7. Suppose we were to find the covariance matrix for these two samples, x_1 and x_2, with 10 features each.

$$x_1 = \begin{pmatrix} 2.5 & 0.5 & 2.2 & 1.9 & 3.1 & 2.3 & 2.0 & 1.0 & 1.5 & 1.1 \end{pmatrix}$$

$$x_2 = \begin{pmatrix} 2.4 & 0.7 & 2.9 & 2.2 & 3.0 & 2.7 & 1.6 & 1.1 & 1.6 & 0.9 \end{pmatrix}$$

$$\begin{aligned} S = \begin{pmatrix} Cov(x_1, x_1) & Cov(x_1, x_2) \\ Cov(x_2, x_1) & Cov(x_2, x_2) \end{pmatrix} &= \begin{pmatrix} Var(x_1) & Cov(x_1, x_2) \\ Cov(x_2, x_1) & Var(x_2) \end{pmatrix} \\ &= \begin{pmatrix} 0.6165555 & 0.6144444 \\ 0.6154444 & 0.7165555 \end{pmatrix} \end{aligned}$$

Theorem 2.8 (Spectral Theorem)**.** *Let $A \in S^n$ be a symmetric Matrix. There exists a symmetric eigenvalue decomposition*

$$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T = U \Lambda U^T, \Lambda = diag(\lambda_1, ..., \lambda_n),$$

where the matrix of $U := [u_1, ..., u_n]$ is orthogonal and contains the eigenvectors of A, while the diagonal matrix Λ contains the eigenvalues of A.

(Important PCA takeaway: symmetric matrix can be factorized into: $S = QDQ^T$ where Q is a matrix of orthonormal eigenvectors of S and D is a diagonal matrix of eigenvalues.)

3 Principal Component Analysis Algorithm

To start, we will have a set of $\{x_n\}$ $n = 1, ..., N$ vectors with dim D. These vectors represent each individual sample with D characteristics.

We want to project the points onto a space that preserves the greatest variance. The goal of PCA is to map the set of vectors to dim M; $M < D$

$$Proj_{u_1}(x_i) = u_1^T x_i u$$

$$\text{Mean of Projections} = u_1^T \bar{x} u$$

Theorem 3.1. *The maximum variance of projections is equal to the largest eigenvalue of the covariance matrix of the mean of projections [6].*

$$u_1^T \mathcal{S} u_1 = \lambda$$

For the above, u_1 corresponds to the unit vector, $\mathcal{S}$ is the covariance matrix, and λ is an eigenvalue of $\mathcal{S}$.

Proof. Computing the Variance of Projections:

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^{N} (u_1^T x_n - u_1^T \bar{x})^2 &= \frac{1}{N} \sum_{n=1}^{N} \left[u_1^T (x_n - \bar{x}) \right]^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} u_1^T (x_n - \bar{x})(x_n - \bar{x})^T u_1 \\ &= u_1^T \left[\frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T \right] u_1 \\ &= u_1^T \mathcal{S} u_1 \qquad \text{S is the covariance matrix} \end{aligned}$$

In order to maximize the variance of projections, we must use Lagrange Multipliers with the restriction

$$u_1^T u_1 = 1$$

$$\begin{aligned}
\frac{d}{du_1}\left[u_1^T \mathcal{S} u_1 + \lambda(1 - u_1^T u_1)\right] &= 0 \\
\frac{d}{du_1}\left[u_1^T \mathcal{S} u_1 + \lambda - \lambda u_1^T u_1\right] &= 0 \\
\cancel{2}\mathcal{S}u_1 - \lambda\cancel{2}u_1 &= 0 \qquad \text{(from derivative of a matrix)} \\
\mathcal{S}u_1 &= \lambda u_1 \\
u_1^T \mathcal{S} u_1 &= \lambda
\end{aligned}$$

□

The eigenvector corresponding to the largest eigenvalue is our first principal component, x1. The next M-1 greatest eigenvalues will be used to find the next M-1 corresponding eigenvectors, resulting in an M dimensional space. Thanks to the Spectral Theorem, we are guaranteed to find an orthogonal basis since the covariance matrix is symmetric.

4 Implementing PCA

To work out PCA, we will apply it to a 2 dimensional data set in R^{10}:

$$x_1 = \begin{pmatrix} 2.5 & 0.5 & 2.2 & 1.9 & 3.1 & 2.3 & 2.0 & 1.0 & 1.5 & 1.1 \end{pmatrix}$$

$$x_2 = \begin{pmatrix} 2.4 & 0.7 & 2.9 & 2.2 & 3.0 & 2.7 & 1.6 & 1.1 & 1.6 & 0.9 \end{pmatrix}$$

The goal of PCA is to minimize the dimension of the data point onto fewer projection vectors in order to easily visualize the dataset as a whole. This will come useful when you are working with high dimensional data.
In our case, we will be reducing the dimension of the dataset to 1.

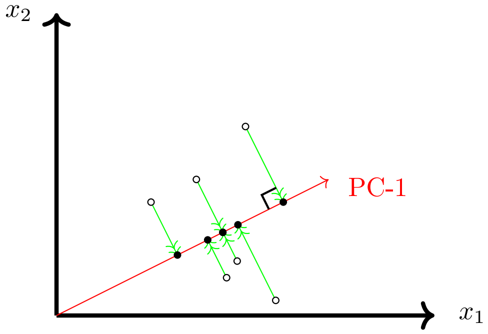

Figure 2: Projecting data points onto principal component vector

1. Data is placed in a matrix, D, with characteristics running down the rows, and observations running across the columns.

$$D = \begin{pmatrix} 2.5 & 0.5 & 2.2 & 1.9 & 3.1 & 2.3 & 2.0 & 1.0 & 1.5 & 1.1 \\ 2.4 & 0.7 & 2.9 & 2.2 & 3.0 & 2.7 & 1.6 & 1.1 & 1.6 & 0.9 \end{pmatrix}$$

2. Means of characteristics are subtracted from the variables and a new matrix, D', is formed.

Mean of first variable:

$$= \frac{1}{10}(2.5 + 0.5 + 2.2 + 1.9 + 3.1 + 2.3 + 2.0 + 1.0 + 1.5 + 1.1) = 1.81$$

Mean of second variable:

$$= \frac{1}{10}(2.4 + 0.7 + 2.9 + 2.2 + 3.0 + 2.7 + 1.6 + 1.1 + 1.6 + 0.9) = 1.91$$

Subtracting means from D:

$$D' = \begin{pmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{pmatrix}$$

3. Compute covariance matrix.
   From Remark 1, the covariance matrix S from our dataset is

$$S = \begin{pmatrix} 0.6165555 & 0.6144444 \\ 0.6154444 & 0.7165555 \end{pmatrix}$$

4. Find eigenvectors of C and place in order of size of eigenvalues.
   For our covariance matrix, the eigenvalues are :

$$0.04908339 \text{ and } 1.2840277,$$

with the corresponding eigenvectors:

$$\begin{pmatrix} -0.7351786 \\ 0.6778733 \end{pmatrix} \text{ and } \begin{pmatrix} 0.6778733 \\ 0.7351786 \end{pmatrix}$$

5. Remove eigenvectors that correspond with eigenvalues who's sum is less than 90% of the total. This step is typical when choosing principal components at the end, where you are left with components that carry the most variation in the data.

6. Order the eigenvectors by their corresponding eigenvalue, from smallest to largest and place them side-by-side in a square matrix W.

$$W = \begin{pmatrix} 0.6778733 & -0.7351786 \\ 0.7351786 & 0.6778733 \end{pmatrix}$$

$$W^T = \begin{pmatrix} 0.6778733 & 0.7351786 \\ -0.7351786 & 0.6778733 \end{pmatrix}$$

Now we have a set of eigenvectors that form a matrix that align with the greatest variation of data.

7. Data expressed in new coordinate system: $D_{pca} = W^T D'$

$$D_{PCA} = \begin{pmatrix} 0.6778733 & 0.7351786 \\ -0.7351786 & 0.6778733 \end{pmatrix} \begin{pmatrix} 0.69 & -1.31 & 0.39 & 0.09 & 1.29 & 0.49 & 0.19 & -0.81 & -0.31 & -0.71 \\ 0.49 & -1.21 & 0.99 & 0.29 & 1.09 & 0.79 & -0.31 & -0.81 & -0.31 & -1.01 \end{pmatrix}$$
$$= \begin{pmatrix} 0.83 & -1.8 & 0.99 & 0.27 & 1.8 & 0.91 & -0.099 & -1.1 & -0.44 & -1.2 \\ -0.18 & 0.14 & 0.38 & 0.13 & -0.21 & 0.18 & -0.35 & 0.046 & 0.018 & -0.16 \end{pmatrix}$$

We are now left with two principal components where we can graph on a coordinate system, row 1 being principal component 1, and row 2 being principal component 2.

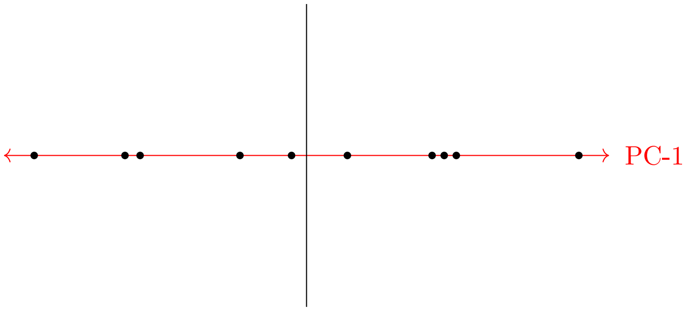

Figure 3: New coordinate system with principal component 1

5 PCA Application: Eigenfaces

In this section, we will walk through the general procedure of Eigenfaces using PCA. Since the dataset will consist of greyscale images of faces, dimensionality reduction will be extremely useful here because we will be working with thousands of data points.

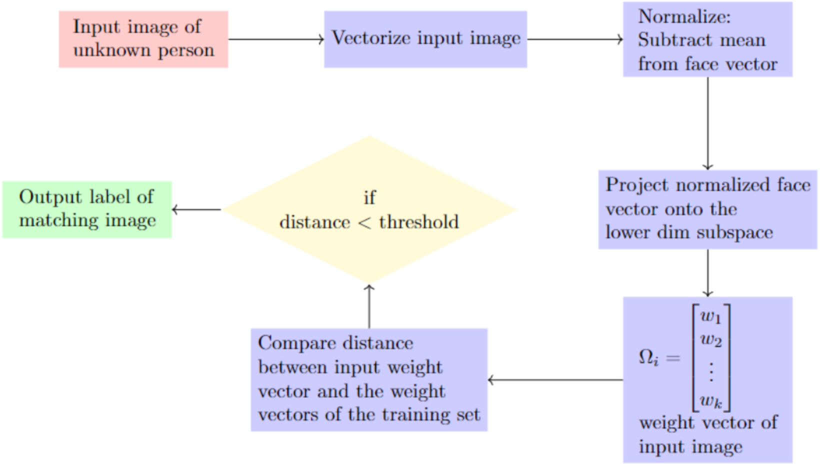
Input image of unknown person
Vectorize input image
Normalize: Subtract mean from face vector
Project normalized face vector onto the lower dim subspace
$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$ weight vector of input image
Compare distance between input weight vector and the weight vectors of the training set
if distance < threshold
Output label of matching image

Figure 4: Eigenface Pseudo-code [3]

1. The first step of eigenface is to collect an array M of grey-scale images of human faces, all in the same pixel dimension $(N \times N)$. This will be our training set. The total data size is MN^2.
2. Now we vectorize each image into $N^2 \times 1$ vectors and can represent the entire dataset as an $N^2 \times M$ matrix. We can call this matrix A, where each column of A represents an image.
3. Same as the original PCA algorithm, we normalize each vector by subtract the mean from each vector in A. This is essentially taking the average face from the data set, and subtracting it from each face, granting us the unique aspects of each face. This set of vectors spans the face vector space.
4. Before we calculate the eigenvectors from the covariance matrix, we must project the face vector space to a lower dimensional subspace and compute the new covariance matrix. The new covariance matrix is obtained by $A^T A$.

5. Now we calculate the eigenvalues and eigenvectors of the covariance matrix C, granting us M eigenvectors ($M \times 1dim$),ordered from greatest to least, also known as eigenfaces. Each eigenface can be used to represent existing and new faces.

6. The next step is to select the $K \leq M$ best eigenfaces that can represent the whole training set.

7. Now we convert the K eigenfaces back to their original face dimensionality.

$$u_i = Av_i$$

Here, u_i is the corresponding $N^2 \times 1$ face vector with the $v_{i^{th}}$ $M \times 1$ eigenvector

8. Now each orignal image can be represented by a linear combination of the K eigenfaces plus the mean face.

$$\begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{bmatrix}_{I_i} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{bmatrix}_{\mu} + w_1 \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{bmatrix}_{ef_1} + w_2 \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_N \end{bmatrix}_{ef_2} + ... + w_K \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{bmatrix}_{ef_K}$$

9. Finally, we can extract the weights and form weight vectors, corresponding to every original face image.

$$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$

Remark 5.1. Suppose our data set is composed of 30 images, 50 × 50 pixels. The total size of our data is $30 \times (50 \times 50) = 75,000$. If we choose K eigenfaces to be 8, (total size = 20,000), we have significantly reduced the amount of data needed to represent the original set.

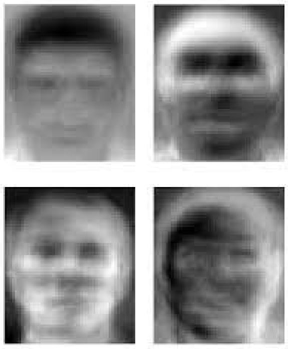

Figure 5: An image of four eigenfaces

References

[1] Clark, Neil, and Ma'ayan, Avi. ”Introduction to Statistical Methods to Analyze Large Data Sets: Principal Components Analysis.” Science Signaling, 2011.

[2] Fuente, Angel de la (2000). Mathematical Methods and Models for Economists. Cambridge: Cambridge University Press. p. 285

[3] ”How PCA Recognizes Faces - Algorithm In Simple Steps.” YouTube, uploaded by Mahvish Nasir, 18 Apr. 2012.

[4] Jolliffe, Ian T.; Cadima, Jorge (2016-04-13). ”Principal component analysis: a review and recent developments”. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 374 (2065): 20150202.

[5] L. Sirovich; M. Kirby (1987). ”Low-dimensional procedure for the characterization of human faces”. Journal of the Optical Society of America A. 4 (3): 519–524.

[6] ”Principal Component Analysis (The Math) : Data Science Concepts.” YouTube, uploaded by ritvikmath, 23 Sept. 2019,

[7] Turk, Matthew A; Pentland, Alex P (1991). Face recognition using eigenfaces (PDF). Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591.