Notes on Principle Component Analysis

Paul F. Roysdon, Ph.D.

I. Introduction

Principle Component Analysis (PCA) is a simple method to perform dimension reduction of large datasets. In fact, PCA can be derived using only basic knowledge of linear algebra.

Suppose we have a large collection of points, $\mathbf{x} = \{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^m$, and we wish to perform lossy compression to these points; i.e. we wish to use less memory to store the data with minimal loss to precision. By PCA we can compute the mean and covariance of the data to determine the most relevant information to retain, discarding the rest.

Computing the *principle components* requires either Eigenvalue Decomposition (EVD) or Singular Value Decomposition (SVD). While both methods work well, we will focus on the EVD. Conveniently, principal components are the eigenvectors of the covariance matrix.

II. Derivation

A. Review of Mean & Covariance

Let the input

$$\mathbf{x} = \{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^m$$

The mean of $\mathbf{x}$ is

$$\begin{aligned}\boldsymbol{\mu}_{\mathbf{x}} &= \mathrm{E}\left\langle \mathbf{x} \right\rangle \\ &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k. \qquad (1)\end{aligned}$$

To compute the covariance, $\mathrm{Cov}\left\langle \mathbf{x} \right\rangle \triangleq \mathbf{C}_{\mathbf{x}}$, first subtract the mean, $\boldsymbol{\mu}_{\mathbf{x}}$, from the input, $\mathbf{x}$,

$$\bar{\mathbf{x}} = \mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}. \qquad (2)$$

Then compute the covariance w.r.t. $\mathbf{x}$

$$\begin{aligned}\mathbf{C}_{\mathbf{x}} &= \mathrm{E}\left\langle \bar{\mathbf{x}}\bar{\mathbf{x}}^{\mathsf{T}} \right\rangle \\ &= \mathrm{E}\left\langle (\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^{\mathsf{T}} \right\rangle \\ &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k\mathbf{x}_k^{\mathsf{T}} - \boldsymbol{\mu}_{\mathbf{x}}\boldsymbol{\mu}_{\mathbf{x}}^{\mathsf{T}}. \qquad (3)\end{aligned}$$

See Section V-A for details.

B. Eigenvalue Decomposition

To prepare for Eigenvalue Decomposition (defined in Section V-A, and a key algorithm in PCA!), consider

$$\mathbf{C}_{\mathbf{x}}\mathbf{e}_i = \lambda_i \mathbf{e}_i,$$

where $\mathbf{e}_i$ are the eigenvectors of $\mathbf{C}_{\mathbf{x}}$, and λ_i are the eigenvalues of $\mathbf{C}_{\mathbf{x}}$. By definition, eigenvectors are orthonormal, i.e. the magnitude (length) of each vector equals 1:

$$|\mathbf{e}_i| = 1,$$

and the dot product of any two eigenvectors is zero (they are perpendicular to each other),

$$\mathbf{e}_i \cdot \mathbf{e}_j = 0, \text{ if } i \neq j.$$

Additionally, the eigenvalues are sorted in descending order

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n.$$

Both the fact that the eigenvectors are orthonormal, and the eigenvalues are sorted, will be important in our analysis. Importantly, the eigenvalues are the “principle components” in the PCA method.

C. Computing the Principle Components

From Eq. 9 in Section V-A, we can reformulate our problem into the EVD form. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be an orthogonal matrix (see Section V-A), whose rows are the eigenvectors of $\mathbf{C}_{\mathbf{x}}$

$$\mathbf{A} = \begin{bmatrix} \mathbf{e}_1^{\mathsf{T}} \\ \mathbf{e}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{e}_n^{\mathsf{T}} \end{bmatrix},$$

such that $\mathbf{A}$ is a transformation matrix, or function mapping, $\mathbf{x} \mapsto \mathbf{y}$, where

$$\begin{aligned}\mathbf{y} &= \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) \qquad (4) \\ &= \mathbf{A}\bar{\mathbf{x}}.\end{aligned}$$

To compute the covariance, $\mathrm{Cov}\left\langle \mathbf{y} \right\rangle \triangleq \mathbf{C}_{\mathbf{y}}$, first subtract the mean, $\boldsymbol{\mu}_{\mathbf{y}}$,

$$\bar{\mathbf{y}} = \mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}},$$

then compute the covariance w.r.t. $\mathbf{y}$

$$\begin{aligned}\mathbf{C}_{\mathbf{y}} &= \mathrm{E}\left\langle \bar{\mathbf{y}}\bar{\mathbf{y}}^{\mathsf{T}} \right\rangle \\ &= \mathrm{E}\left\langle (\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})(\mathbf{y} - \boldsymbol{\mu}_{\mathbf{y}})^{\mathsf{T}} \right\rangle. \qquad (5)\end{aligned}$$

Evaluating Eqn. 4, $\boldsymbol{\mu}_{\mathbf{y}} = \mathbf{0}$, because

$$\begin{aligned}\mathrm{E}\left\langle \mathbf{y} \right\rangle &= \mathrm{E}\left\langle \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) \right\rangle \\ &= \mathbf{A}\left(\mathrm{E}\left\langle \mathbf{x} \right\rangle - \mathrm{E}\left\langle \boldsymbol{\mu}_{\mathbf{x}} \right\rangle\right) \\ &= \mathbf{0}. \qquad (6)\end{aligned}$$

Applying Eqn. 6 to Eqn. 5, $\mathbf{C_y}$ simplifies to

$$\begin{aligned}\mathbf{C_y} &= \mathrm{E}\left\langle \bar{\mathbf{y}}\bar{\mathbf{y}}^\intercal \right\rangle \\ &= \mathrm{E}\left\langle \{\mathbf{A}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})\}\{\mathbf{A}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})\}^\intercal \right\rangle \\ &= \mathrm{E}\left\langle \mathbf{A}(\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})(\mathbf{x}-\boldsymbol{\mu}_\mathbf{x})^\intercal\mathbf{A}^\intercal \right\rangle \\ &= \mathbf{A}\mathrm{E}\left\langle \mathbf{x}\mathbf{x}^\intercal \right\rangle \mathbf{A}^\intercal \\ &= \mathbf{A}\mathbf{C_x}\mathbf{A}^\intercal. \end{aligned} \tag{7}$$

Notice that $\mathbf{C_y}$ is a function of $\mathbf{C_x}$ and $\mathbf{A}$. Also notice that Eqn. 7 is now in the form of eqn. 9.

Now consider the analysis of $\mathbf{C_y} = \mathbf{A}\mathbf{C_x}\mathbf{A}^\intercal$. We have

$$\mathbf{A} = \begin{bmatrix} \mathbf{e}_1^\intercal \\ \mathbf{e}_2^\intercal \\ \vdots \\ \mathbf{e}_n^\intercal \end{bmatrix}, \text{ and } \mathbf{A}^\intercal = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \ldots & \mathbf{e}_n \end{bmatrix}.$$

Recall that the columns of $\mathbf{A}^\intercal$ are the eigenvectors of $\mathbf{C_x}$, and multiplication of $\mathbf{C_x}\mathbf{A}^\intercal$ are the eigenvectors times the eigenvalues

$$\begin{aligned}\mathbf{C_x}\mathbf{A}^\intercal &= \mathbf{C_x}\begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \ldots & \mathbf{e}_n \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1\mathbf{e}_1 & \lambda_2\mathbf{e}_2 & \ldots & \lambda_n\mathbf{e}_n \end{bmatrix}.\end{aligned}$$

Then it is obvious that

$$\begin{aligned}\mathbf{C_y} &= \mathbf{A}\mathbf{C_x}\mathbf{A}^\intercal \\ &= \begin{bmatrix} \mathbf{e}_1^\intercal \\ \mathbf{e}_2^\intercal \\ \vdots \\ \mathbf{e}_n^\intercal \end{bmatrix} \begin{bmatrix} \lambda_1\mathbf{e}_1 & \lambda_2\mathbf{e}_2 & \ldots & \lambda_n\mathbf{e}_n \end{bmatrix} \\ &= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & 0 \\ \vdots & & \ddots & \\ 0 & 0 & & \lambda_n \end{bmatrix},\end{aligned}$$

where λ_i are the eigenvalues of $\mathbf{C_x}$, and are called the *principle components*.

By analysis, given the mapping in Eqn. 4, we can express any estimate $\hat{\mathbf{x}}$ of the original vector $\mathbf{x}$ as a linear combination of the principle components

$$\begin{aligned}\mathbf{y} &= \mathbf{A}(\hat{\mathbf{x}}-\boldsymbol{\mu}_\mathbf{x}) \\ \mathbf{A}^\intercal\mathbf{y} &= \mathbf{A}^\intercal\mathbf{A}(\hat{\mathbf{x}}-\boldsymbol{\mu}_\mathbf{x}) \\ \hat{\mathbf{x}} &= \mathbf{A}^\intercal\mathbf{y}+\boldsymbol{\mu}_\mathbf{x}.\end{aligned}$$

Therefore the mapping of the first k principle components is

$$\hat{\mathbf{x}} = \mathbf{A}_k^\intercal\mathbf{y}+\boldsymbol{\mu}_\mathbf{x}. \tag{8}$$

III. Examples

A. Simple 2D Problem

Given a dataset $\mathbf{x}$ (see Fig. 1), compute the standard deviation in two dimensions, (σ_1^2, σ_2^2). This requires two steps.

First, compute the PCs using PCA. This will force the data to be zero-mean, and create axes (components) from which to calculate the standard deviation. From eqns. 1, 2, 3, 9, and 4, the PCA method is summarized in just five lines of code!

```
1 mu_x = mean(x);
2 x_bar = x — mu_x;
3 C_x = cov(x_bar);
4 [V,D] = eig(C_x);
5 y = [V(:,2)'; V(:,1)']*x_bar';
```

Line 4 of the above code-block produces the eigenvectors shown in Fig. 2, while line 5 rotates the data into the axes necessary for analysis (shown in Fig. 3).

Next, perform the analysis. Using the first PC, we can calculate the standard deviation σ_1^2, as shown in the code-block below, and graphically in Fig. 4. Note the first PC corresponds to the variation in the y-direction – after the data is rotated into the new axes – and the second PC corresponds to the variation in the x-direction.

```
y_1 = [V(:,1)']*x_bar';
sigma_1 = std(y_1);
```

B. Image Reconstruction

Given the Stanford database of prominent faces, shown in Fig. 5, perform lossy compression so that the shapes in the image are retained and recognizable (the term "recognizable" is subjective).

First, compute the PCs using PCA, as shown in the previous example using eqns. 1, 2, 3, 9, and 4.

Next perform image reconstruction using only 1, 10, 100, or all principle components using eqn. 8. The results are shown in Figs. 6-9.

C. Image Recognition

Given a photo database of faces, shown in Fig. 10, perform facial recognition (matching). In the literature, this example is commonly named "EigenFaces".

First, compute the PCs using PCA, as shown in the previous examples. The mean of all faces in the database, $\boldsymbol{\mu}_f$, (recall eqn. 1) is shown in Fig. 11. The 90 eigenvectors, $\mathbf{V}_f$, (recall eqn. 9) are presented in Fig. 12. Note that only the first 20 eigenvectors are large enough for further analysis.

Next perform facial recognition using the PCA results. Compute a facial "signature", $\gamma_d \in \mathbb{R}^m$, for each image in the database:

$$\gamma_d = \boldsymbol{\mu}_f^\intercal\mathbf{V}_f.$$

For the face shown in the upper left of Fig. 13, compute the new face "signature", γ_n:

$$\gamma_n = \boldsymbol{\mu}_n^\intercal\mathbf{V}_f.$$

Next, compute the Euclidean distance (L_2-norm) of the database signatures versus the new face signature,

$$\boldsymbol{\delta}_n = \|\gamma_d - \gamma_n\|_2.$$

Finally, sort $\boldsymbol{\delta}_n$ in ascending order. The "best match" is the row of $\boldsymbol{\delta}_n$ with the smallest Euclidean distance (smallest numerical value). The best match, as well as 2^{nd} and 3^{rd} best estimates are shown in the second row of Fig. 13.

IV. Final Comments

This paper presented a simple linear algebra derivation of PCA, with key equations translated into Matlab code. Complete Matlab demos are provided with this paper so that the user can gain practical experience using PCA and reuse the code as a recipe for future projects.

Comments: PCA decorrelates multivariate data, finds useful components, reduces dimensionality, and performs non-parametric analysis, e.g. no parameters or coefficients to adjust.

Limitations: By design, PCA will not work on non-linear data. Consider *kernel PCA* instead.

Finally, remember that PCA is only the first step of data analysis.

V. Appendix

A. Definitions & Theorems

Definition 5.1: Matrix Symmetry: A matrix $\mathbf{A}$ is symmetric *iff* $\mathbf{A} = \mathbf{A}^\intercal$. ♠

Definition 5.2: Orthogonal diagonalizable: Let $\mathbf{A}$ be an $n \times n$ matrix. $\mathbf{A}$ is orthogonal diagonalizable if there is an orthogonal matrix $\mathbf{S}$ such that $\mathbf{S}^{-1}\mathbf{A}\mathbf{S}$ is diagonal. ♠

Theorem 5.3: Spectral Theorem: Let $\mathbf{A}$ be an $n \times n$ matrix. $\mathbf{A}$ is orthogonal diagonalizable *iff* $\mathbf{A}$ is symmetric. ♠

Definition 5.4: Covariance: Let $\mathrm{E}\left\langle u \right\rangle$ be the mean of a the random variable u. Then the covariance $\mathrm{Cov}\left\langle x, y \right\rangle$ of random variables x, y is defined as $\mathrm{Cov}\left\langle x, y \right\rangle = \mathrm{E}\left\langle xy \right\rangle - \mathrm{E}\left\langle x \right\rangle \mathrm{E}\left\langle y \right\rangle$. ♠

Definition 5.5: Variance: The variance of a random variable x is $\mathrm{Var}\left\langle x \right\rangle = \mathrm{Cov}\left\langle x, x \right\rangle$. ♠

Definition 5.6: Covariance Matrix: The covariance matrix $\mathbf{\Sigma}$ of $\mathbf{X}$ is the matrix with entries $\mathbf{\Sigma}_{i,j} = \mathrm{Cov}\left\langle x_i, x_j \right\rangle$. ♠

Definition 5.7: Eigenvalue Decomposition (EVD): Two forms of the Eigenvalue Decomposition exist.

1) Symmetric Square decomposed into squares: Assume $\mathbf{A}$ to be $n \times n$ and symmetric. Then

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^\intercal \tag{9}$$

where $\mathbf{D}$ is diagonal with the eigenvalues of $\mathbf{A}$, and $\mathbf{V}$ is orthogonal and the eigenvectors of $\mathbf{A}$.

2) Square decomposed into squares: Assume $\mathbf{A} \in \mathbb{R}^{n \times n}$. Then

$$\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{U}^\intercal$$

where $\mathbf{D}$ is diagonal with the square root of the eigenvalues of $\mathbf{A}\mathbf{A}^\intercal$, $\mathbf{V}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^\intercal$ and $\mathbf{U}^\intercal$ are the eigenvectors of $\mathbf{A}^\intercal\mathbf{A}$.

♠

Definition 5.8: Inverse & Transpose: Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be an orthogonal matrix where $\mathbf{a}_i$ is the i^{th} column vector. The i, j^{th} element of $\mathbf{A}^\intercal\mathbf{A}$ is

$$(\mathbf{A}^\intercal\mathbf{A})_{ij} = \mathbf{a}_i^\intercal\mathbf{a}_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, because $\mathbf{A}^\intercal\mathbf{A} = \mathbf{I}_n$, it follows that $\mathbf{A}^{-1} = \mathbf{A}^\intercal$. ♠

Definition 5.9: Properties of a Symmetric Matrix: For any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, then $\mathbf{A}^\intercal\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}\mathbf{A}^\intercal \in \mathbb{R}^{m \times m}$ are square and symmetric:

$$(\mathbf{A}\mathbf{A}^\intercal)^\intercal = \mathbf{A}\mathbf{A}^\intercal$$
$$(\mathbf{A}^\intercal\mathbf{A})^\intercal = \mathbf{A}^\intercal\mathbf{A}.$$

♠

B. Example Results & Figures

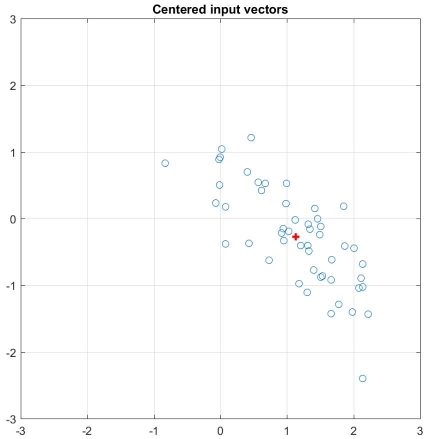

Fig. 1. Input data and mean $\boldsymbol{\mu}_{\mathbf{x}}$.

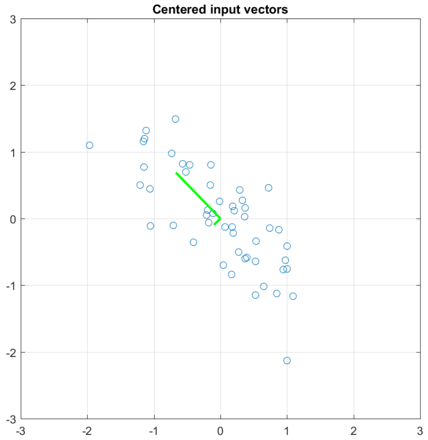

Fig. 2. Principle Component vectors.

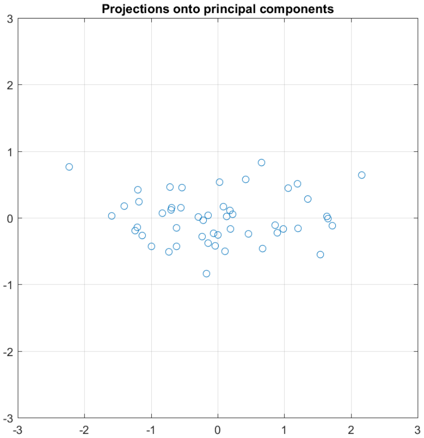

Fig. 3. Projections onto Principle Components.

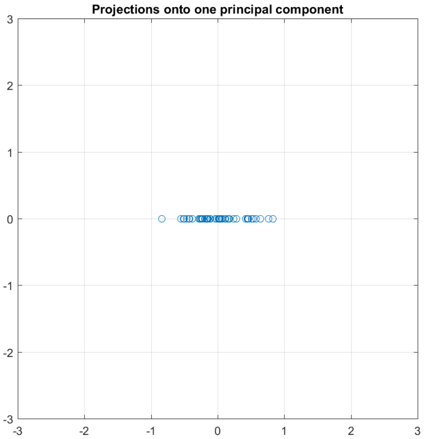

Fig. 4. Projections onto 1 Principle Component.

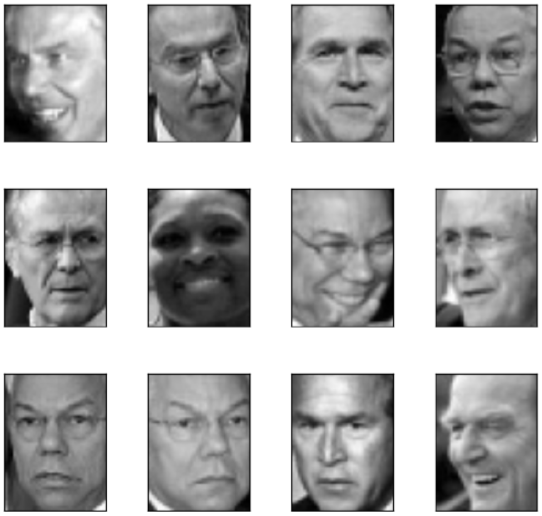

Fig. 5. Original Stanford dataset of faces.

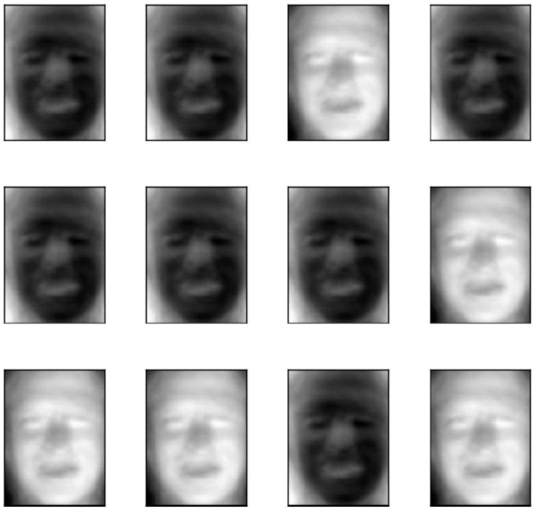

Fig. 6. Stanford dataset using only 1 PC.

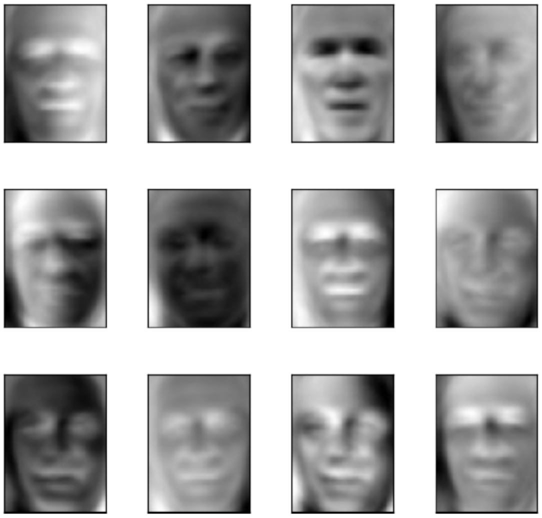

Fig. 7. Stanford dataset using 10 PCs.

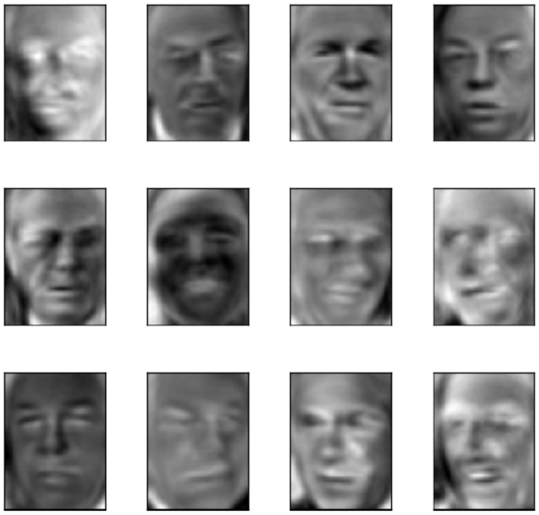

Fig. 8. Stanford dataset using 100 PCs.

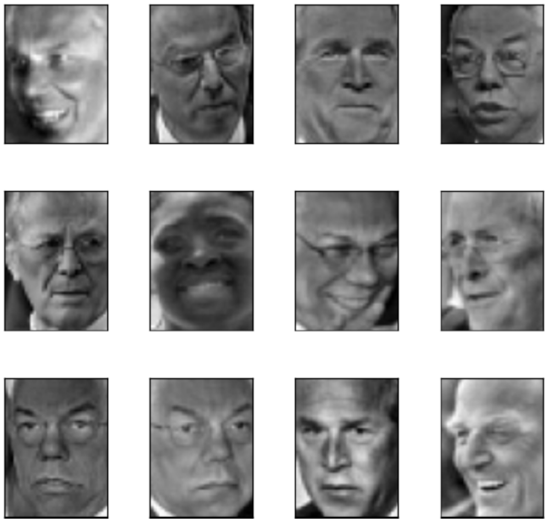

Fig. 9. Stanford dataset using all PCs.

Fig. 10. Partial AT&T dataset of faces.

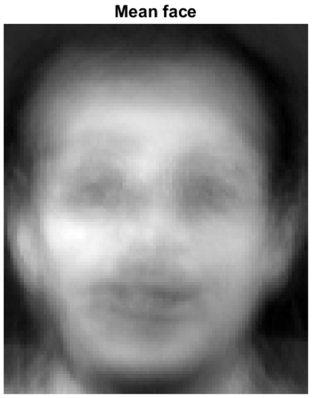

Fig. 11. Mean of all faces in the AT&T dataset.

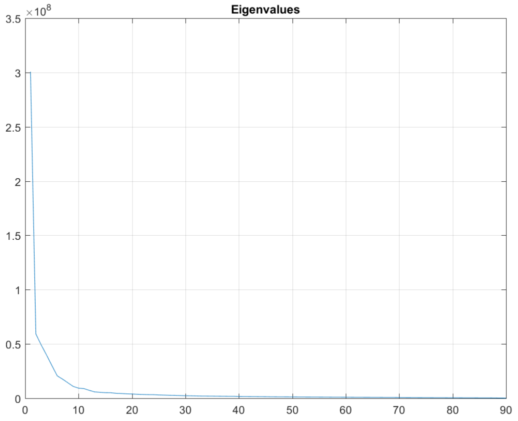

Fig. 12. Eigenvalues 1-90 for the AT&T dataset.

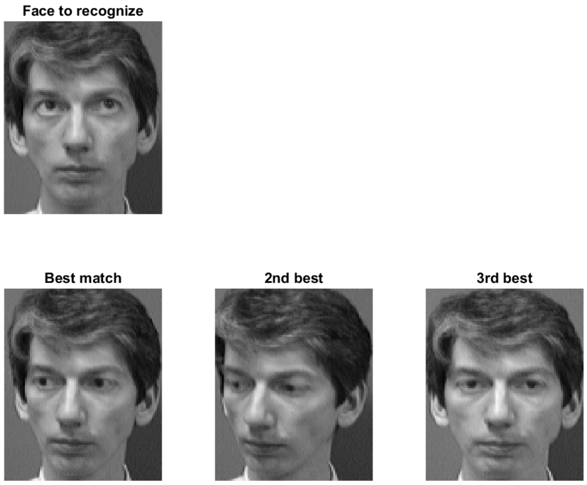

Fig. 13. Prediction (best match with smallest Euclidean distance from the mean), as well as 2^{nd} and 3^{rd} best estimates.