Recall the centered data matrix
\[ \tilde{X} := [\tilde{X}_1, \ldots, \tilde{X}_n] \in \mathbb{R}^{d \times n} \]
\[ \tilde{X}_j := X_j - \bar{X}, \quad \bar{X} := \frac{1}{n} \sum_{j=1}^{n} X_j, \]
and the sample covariance matrix
\[ S := \frac{1}{n} \tilde{X} \tilde{X}^T \]

Then, PCA is nothing but the eigenvalue decomposition of \( S \)
\[ S = \Phi \Lambda \Phi^T, \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d) \]
\[ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0. \]
\[ \Phi := [\phi_1, \ldots, \phi_d] \in \mathbb{R}^{d \times d} \]

is an orthonormal matrix, and \( \{\phi_1, \ldots, \phi_d\} \)
form an ONB of \( \mathbb{R}^d \).
\[ \phi_j^T \tilde{X} \]

is said to be the \( j \)th

principal components of \( \tilde{X} \).

These are nothing but the expansion coefficients of \( \tilde{X} \) w.r.t. the ONB vector \( \phi_j \).

If \( \tilde{X} \) forms a “cigar” shape,

then \( \phi_j^T \tilde{X} \) are the coordinate values of \( \tilde{X} \)

under the rotated axes.
• Hence viewing the given dataset under the principal axes \( \phi_1, \phi_2, \ldots \) provides us better interpretations of the data than viewing them under the original axes \( e_1, e_2, \ldots \).

• PCA is also often used as a tool to do dimension reduction and feature extraction by keeping only top \( k \) PCA coordinates where \( k \ll d \), i.e.,

\[
\Phi_k := [\phi_1, \ldots, \phi_k] \in \mathbb{R}^{d \times k}
\]

\( \mathbb{R}^d \ni \tilde{x}_j \mapsto \Phi_k^T \tilde{x}_j \in \mathbb{R}^k \)

\( \text{top k PCA coordinates of } \tilde{x}_j \).

Note that using these top \( k \) principal components, we can approximate the original data \( x_j \) by

\[
x_j \approx \bar{x} + \Phi_k \tilde{x}_j
\]

Of course the approximation gets better and better as \( k \) increases. In fact, if \( k = d \), then \( x_j \) is recovered exactly (within machine \( \varepsilon \)).
Now we'll face the problem when we compute the eigendecomposition of \( S = \Phi \Lambda \Phi^T \):

1. If \( d \) is large, we cannot compute this eigendecomposition because we cannot hold \( \Phi \in \mathbb{R}^{d \times d} \) in computer memory, and its computational cost is \( O(d^3) \), i.e., too expensive to compute.

2. Fortunately, we often do not need all \( d \) eigenvectors, most likely, only first \( k \) eigenvectors \( k \ll d \).

3. Moreover if \( d > n \), then \( \text{rank}(S) = n - 1 \) if \( x_j \)'s are linearly indep. So, after the first \( n-1 \) eigenvectors are useless!

\[
\hat{S} = \frac{1}{n} \hat{X} \hat{X}^T = \frac{1}{n} \left\{ \hat{x}_1 \hat{x}_1^T + \cdots + \hat{x}_n \hat{x}_n^T \right\}
\]

Why? \( \hat{S} = \frac{1}{n} \hat{X} \hat{X}^T = \frac{1}{n} \left\{ \hat{x}_1 \hat{x}_1^T + \cdots + \hat{x}_n \hat{x}_n^T \right\} \)

So looks like \( \text{rank}(S) = n \).

But since \( \sum \hat{x}_i = 0 \) because the mean \( \bar{x} \) is subtracted from each data vector \( x_j \) (i.e., \( \hat{x}_j = x_j - \bar{x} \))

Hence, \( S \) loses 1 rank. So, \( \text{rank}(S) = n - 1 \).
Now, let's consider the reduced SVD of $\tilde{X}$:

$$\tilde{X} = \hat{U} \hat{\Sigma} \hat{V}^T$$

For $d > n$, $\hat{\Sigma}$ is a diagonal matrix with positive entries on the diagonal. For $d < n$, $\hat{\Sigma}$ is a matrix of rank $d$.

Just consider the "neo-classical" setting, i.e., $d \geq n$ (e.g., the face image database).

Then consider the sample covariance matrix $S$ using the above SVD:

$$S = \frac{1}{n} \tilde{X} \tilde{X}^T = \frac{1}{n} \hat{U} \hat{\Sigma} \hat{V}^T \hat{V} \Sigma^T \hat{U}^T$$

$$= \frac{1}{n} \hat{U} \Sigma \Sigma^T \hat{U}^T = \frac{1}{n} \hat{U} \Sigma^2 \hat{U}^T$$

Now $\hat{\Sigma} = \text{diag}(\sigma_1, \ldots, \sigma_{n-1}, 0)$ if $X_1, \ldots, X_n$ are linearly independent.

So, $\hat{\Sigma}^2 = \text{diag}(\sigma_1^2, \ldots, \sigma_{n-1}^2, 0)$.

Finally, $S$ can be written as

$$S = \hat{U} \left( \frac{1}{n} \hat{\Sigma}^2 \right) \hat{U}^T$$

$$\hat{\Sigma}^2 = \text{diag}(\sigma_1^2, \ldots, \sigma_{n-1}^2, 0)$$

columns are orthonormal.

Comparing this with the eigendecomposition.
\[ S = \Phi \Lambda \Phi^T, \text{ we can conclude that} \]
\[
\begin{align*}
\Phi(:, 1:n) &= \hat{U} \\
\Lambda(1:n, 1:n) &= \frac{1}{n} \hat{\Sigma}^2 = \text{diag}(\sigma^2_1, \ldots, \sigma^2_{n-1}, 0)
\end{align*}
\]

In fact, only the 1: n-1 portion is useful since \( \sigma_n = 0 \).

Hence, we should use the reduced SVD of \( \tilde{X} \) (not \( S \)) for computing PCA!!

Do not use the eigendecomposition of \( S \) unless \( d \) is small.

\textbf{Note:} \( \tilde{X} V = \hat{U} \hat{\Sigma} V^T V = \hat{U} \hat{\Sigma} \]
\[
= [\sigma_1 u_1, \ldots, \sigma_{n-1} u_{n-1}, 0]
\]
\[
= [\tilde{X} v_1, \ldots, \tilde{X} v_n]
\]

So, \( u_j = \frac{1}{\sigma_j} \tilde{X} v_j, \quad j = 1, \ldots, n-1. \)

In other words, each principal axis \( u_j \) is just a linear combination of the (centered) input vectors \( \tilde{X}_1, \ldots, \tilde{X}_n \)!

Now let's do MATLAB experiments using the face image database consisting of 143 faces each of which has \( 128 \times 128 = 16384 \) pixels, i.e., \( d = 16384, \ n = 143 \).