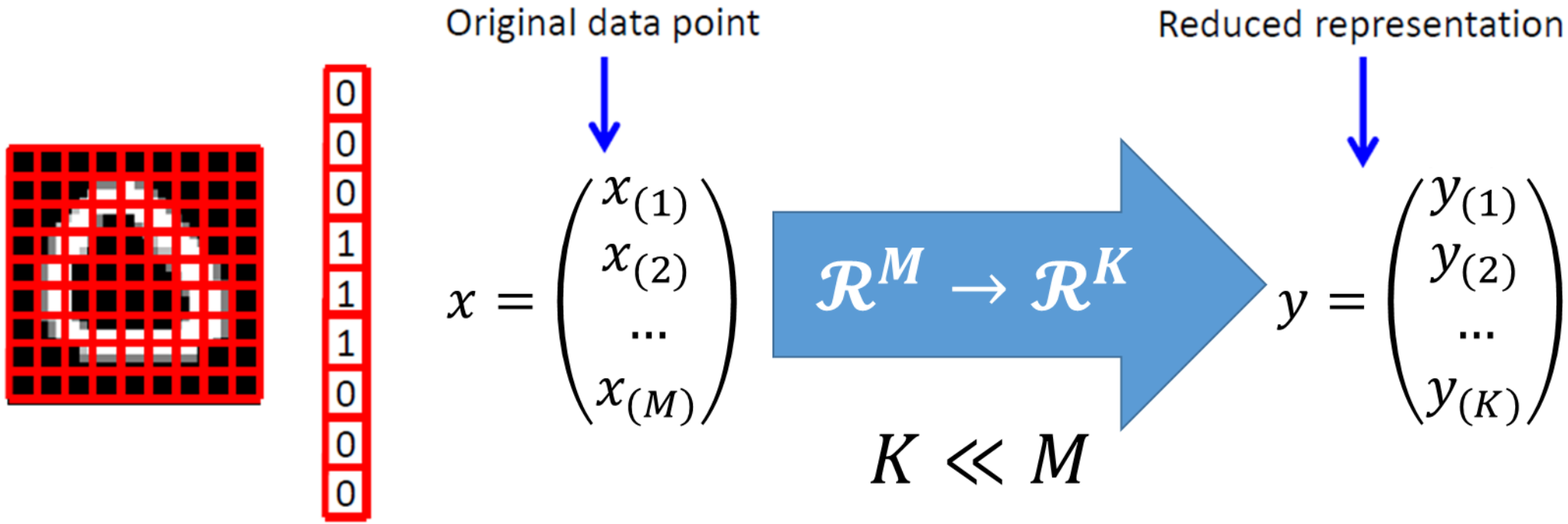
# Principal Component Analysis
## - Dimensionality Reduction and Feature Extraction

Liang Liang

# What is dimensionality reduction (data compression) ?

- The process of reducing the number of variables in a feature vector
  - One can combine, transform or select variables
  - One can use linear or nonlinear operations

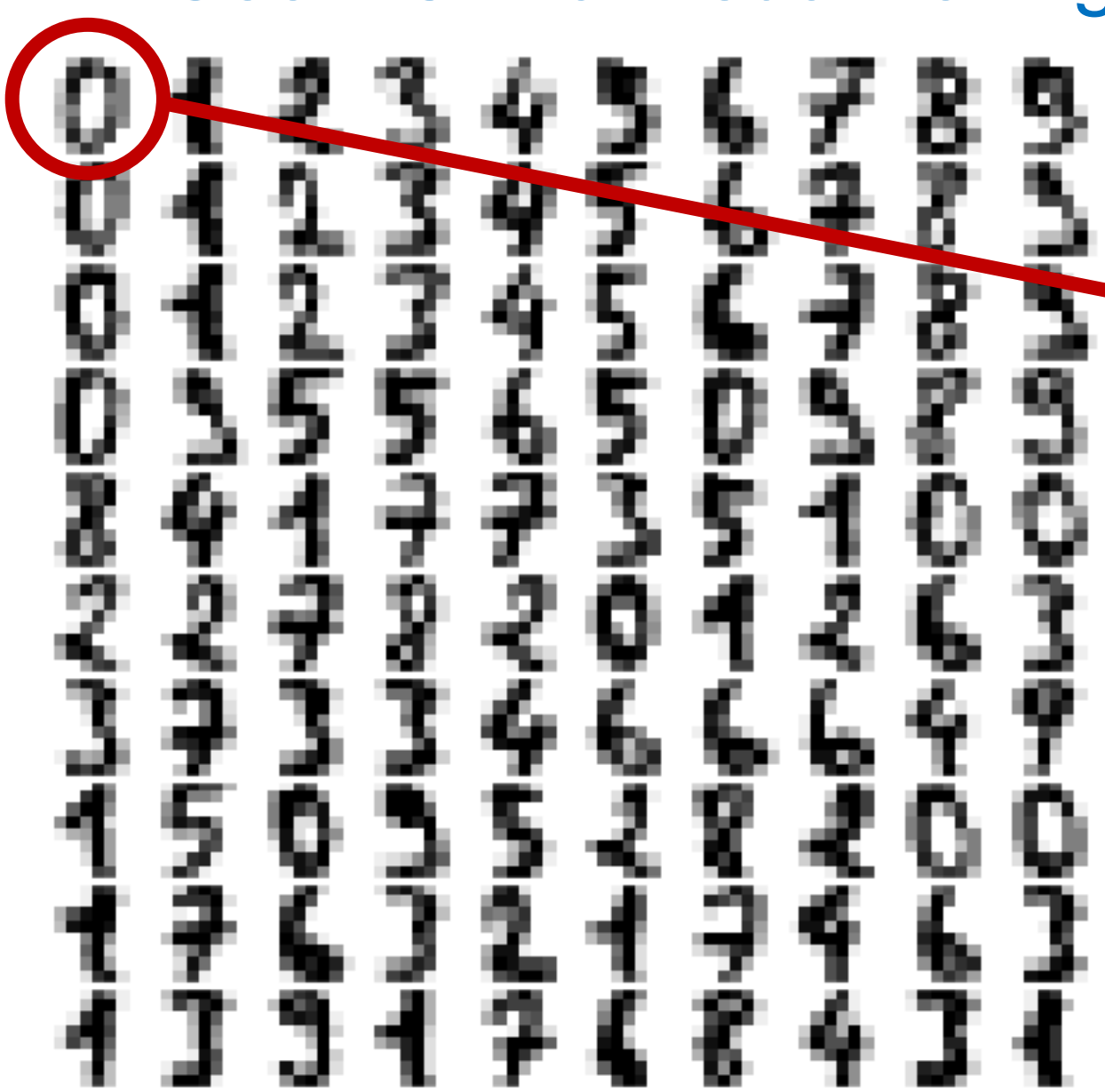


# Why dimensionality reduction ?

- The dimension-reduced data can be used for
  - Visualizing, exploring and understanding the data in 2D or 3D
  - Cleaning the data
  - Speeding up subsequent learning task
  - Building simpler models

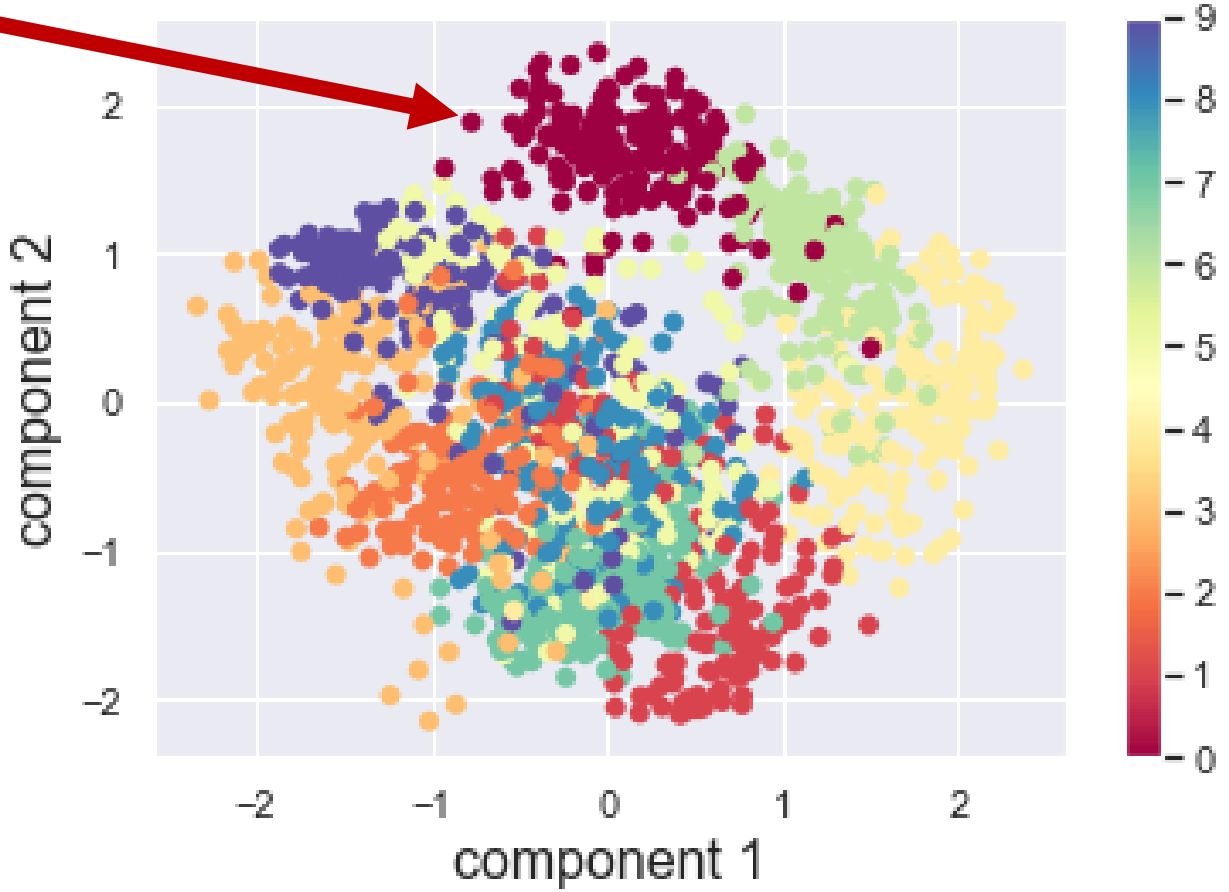
- Key questions of a dimensionality reduction algorithm
  - What is the criterion for carrying out the reduction process?
  - What are the algorithm steps?

# Use PCA to Visualize High Dimensional Data



Each 8-by-8 image has 64 pixels

$$x_n \in \mathcal{R}^{64} \Rightarrow \text{PCA} \Rightarrow y_n \in \mathcal{R}^2$$



see the demo PCA_visualization_denoising.ipynb

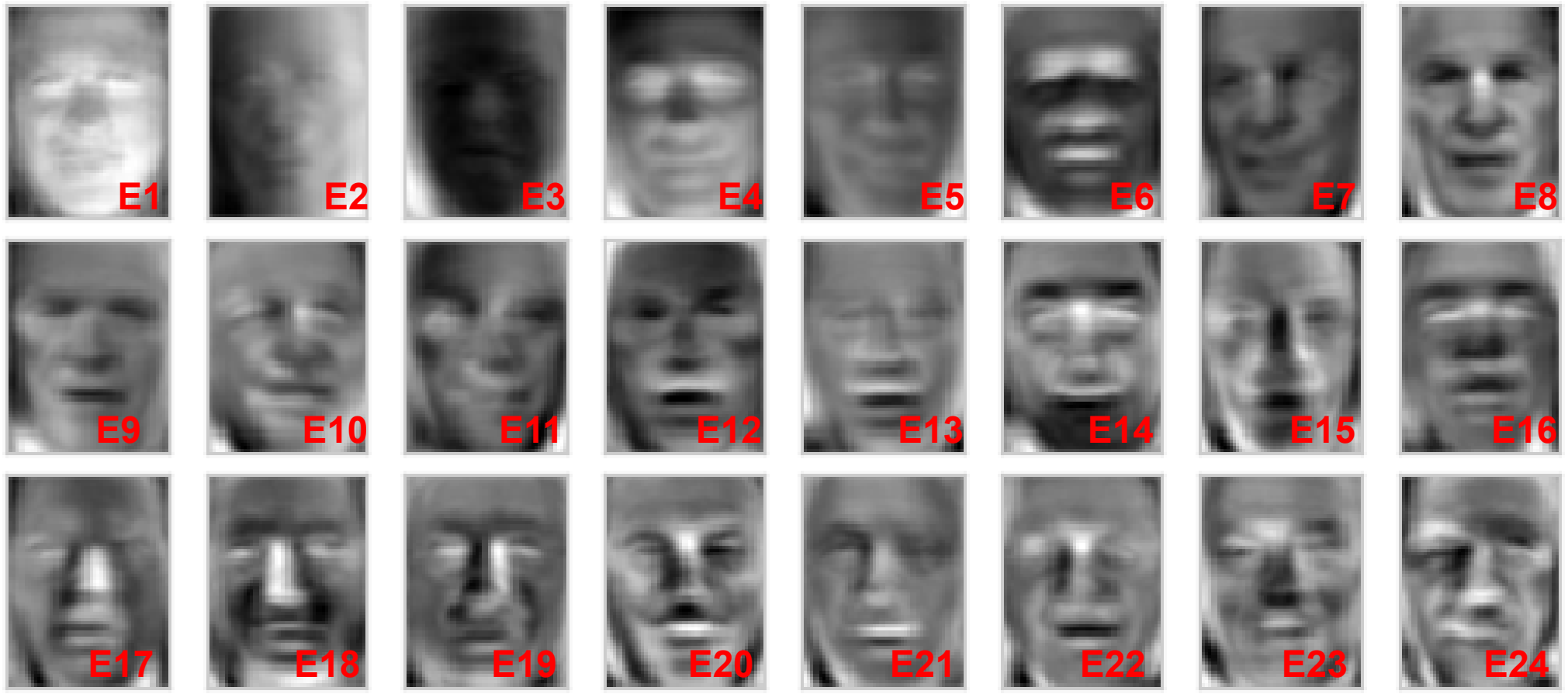
# PCA for Feature Extraction: Eigenfaces

mean face

E0

image size
=62*47=**2914**

**X**




**= E0 + (-0.015)×E1 + (0.0037)×E2 + (-0.01)×E3 + …+…**

**New Feature Vector y = [-0.015, 0.0037, -0.01, …] 24 numbers**

# PCA for data “transform”

- Transform data to hide business secrete
- Why do we need to do this?

Example: https://www.kaggle.com/mlg-ulb/creditcardfraud

*B. Features Augmentation*

Any transaction request is described by few variables such as the merchant ID, cardholder ID, purchase amount, date and time. All transactions requests passing the blocking rules are entered in a database containing all recent authorized transactions, where the feature-augmentation process starts. During feature augmentation, a specific set of *aggregated* features associated to each authorized transactions is computed, to provide additional information about the purchase and better discriminate frauds from genuine transactions. Examples of aggregated features are the average expenditure of the customer every week/month, the average number of transactions per day or in the same shop, the average transaction amount, the location of the last purchases [7], [8], [23], [41], [45], [66]. Van Vlasselaer et al. [63] show that additional informative features can be extracted from the social networks connecting the cardholders with merchants/shops.

- $C \in \mathcal{R}^{M \times M}$ is a symmetric matrix, $\lambda$ is an eigenvalue and $w$ is an eigenvector of $C$ if

$$Cw = \lambda w, \text{ and } w \neq 0$$

  - eigenvalues are real numbers, $\{\lambda_1, \ldots, \lambda_M\}$, $D$=diag($\{\lambda_1, \ldots, \lambda_M\}$)
  - eigenvectors are orthogonal, $\{w_1, \ldots, w_M\}$,

$$w_i{}^T w_j = 0 \text{ if } i \neq j$$

  - $aw_m$(scalar $a > 0$) is also an eigenvector because $Caw_m = \lambda a w_m$

    Usually, we add this constraint $\|w_m\|_2 = \sqrt{w_m{}^T w_m} = 1$

  - If we find $M$ linearly independent eigenvectors of $C$, then

$$P = [w_1, \ldots, w_M], P^{-1} = P^T, \quad P^T P = I$$

  - Using Eigen Decomposition Theorem

$$C = PDP^T$$

- $C \in \mathcal{R}^{M \times M}$ is a symmetric matrix, $\lambda$ is an eigenvalue and $w$ is an eigenvector of $C$ if

$$Cw = \lambda w, \text{ and } w \neq 0$$

  - eigenvalues are real numbers, $\{\lambda_1, \ldots, \lambda_M\}$, $D$=diag($\{\lambda_1, \ldots, \lambda_M\}$)
  - eigenvectors are orthogonal, $\{w_1, \ldots, w_M\}$, $w_m \in \mathcal{R}^M$

$$w_i{}^T w_j = 0 \text{ if } i \neq j$$

  - We add this constraint $\|w_m\|_2 = \sqrt{w_m{}^T w_m} = 1$

- A useful equation: $Cw_m = \lambda_m w_m \Rightarrow w_m{}^T C w_m = \lambda_m w_m{}^T w_m = \lambda_m$

# Principal component analysis (PCA) - the algorithm steps

- Input: $N$ data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$
- Step-1: Compute the mean $\mu$ and covariance matrix $C$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n \qquad C = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)\,(x_n - \mu)^T$$

- Step-2: Compute the eigenvectors $w_1$, $w_2 \ldots, w_M$ of $C$, and the corresponding eigenvalues $\lambda_1, \ldots \lambda_M$ (real, nonnegative)

$$\lambda_1 \geq \lambda_2 \ldots \geq \lambda_M \geq 0$$

# Principal component analysis (PCA) - the algorithm steps

- Input: $N$ data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$
- Step-1: Estimate the mean $\mu$ and covariance matrix $C$ from the data

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n, \qquad C = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)\,(x_n - \mu)^T$$

- Step-2: Compute the eigenvectors $w_1, w_2 \ldots, w_M$ of $C$, and the corresponding eigenvalues $\lambda_1, \ldots \lambda_M$, where $\lambda_1 \geq \lambda_2 \ldots \geq \lambda_M \geq 0$
- Step-3: Compute the reduced representation of data (a new feature vector)

$$\text{vector} \in \mathcal{R}^M \quad x_n \Rightarrow \; y_n = \begin{bmatrix} w_1^T(x_n - \mu)/\sqrt{\lambda_1} \\ w_2^T(x_n - \mu)/\sqrt{\lambda_2} \\ \ldots \\ w_K^T(x_n - \mu)/\sqrt{\lambda_K} \end{bmatrix} \qquad \text{vector} \in \mathcal{R}^K$$

- Output: $N$ new data points $\{y_1, y_2, y_3, \ldots, y_N\}$ and $y_n \in \mathcal{R}^K$, $K \ll M$, $K \ll N$

# PCA: the forward transform from $x_n$ to $y_n$

$x_n \in \mathcal{R}^M$, $y_n \in \mathcal{R}^K$, $x_n$ and $y_n$ are two feature vectors of the same object

$$y_n = A\,(x_n - \mu), \text{ where } A = \begin{bmatrix} w_1^T/\sqrt{\lambda_1} \\ w_2^T/\sqrt{\lambda_2} \\ \dots \\ w_K^T/\sqrt{\lambda_K} \end{bmatrix} \text{ and } \lambda_1 \geq \lambda_2 \dots \geq \lambda_K$$

PCA is a linear transform from $x_n$ to $y_n$

$w_k$ is called a principal axis/direction

an element in $y_n$ is a principal component

$K$ is the number of principal components, $K \ll M$ ($K$ is set by the user)

$A$ is called projection matrix of PCA

dimensionality reduction: select a few principal components (e.g., $K = 3 \ll M$)

# PCA: the forward transform from $x_n$ to $y_n$ or $\beta_n$

- $y_n = [y_{n,1} \quad \cdots \quad y_{n,K}]^T$ where $y_{n,k} = w_k^T (x_n - \mu)/\sqrt{\lambda_k}$
- $\beta_n = [\beta_{n,1} \quad \cdots \quad \beta_{n,K}]^T$ where $\beta_{n,k} = w_k^T (x_n - \mu)$ is a scalar
- $y_n \; and \; \beta_n \in \mathcal{R}^K$ are two "new" feature vectors of the same object

  $y_{n,k} = \beta_{n,k}/\sqrt{\lambda_k}$ and $\beta_{n,k} = y_{n,k}\sqrt{\lambda_k}$

# PCA: the forward transform from $x_n$ to $y_n$ or $\beta_n$

$x_n \in \mathcal{R}^M, y_n \in \mathcal{R}^K, \beta_n \in \mathcal{R}^K$

$x_n$, $y_n$, and $\beta_n$ are feature vectors of the same object

$$\beta_n = W\,(x_n - \mu), \text{ where } W = \begin{bmatrix} w_1^T \\ w_2^T \\ \dots \\ w_K^T \end{bmatrix} \text{ and } \lambda_1 \geq \lambda_2 \dots \geq \lambda_K$$

$$y_n = A\,(x_n - \mu), \text{ where } A = \begin{bmatrix} w_1^T/\sqrt{\lambda_1} \\ w_2^T/\sqrt{\lambda_2} \\ \dots \\ w_K^T/\sqrt{\lambda_K} \end{bmatrix} \text{ and } \lambda_1 \geq \lambda_2 \dots \geq \lambda_K$$

# PCA: the inverse transform from $y_n$ to $\tilde{x}_n$

- $x_n \in \mathcal{R}^M$, $y_n \in \mathcal{R}^K$, $x_n$ and $y_n$ are two feature vectors of the same object
- Inverse transform from $y_n$ to $\tilde{x}_n$ : $\tilde{x}_n = By_n + \mu$
  $B$ is a matrix
- $x_n = \tilde{x}_n$ if and only if $K = M$
- $x_n \approx \tilde{x}_n$ if $K < M$ (usually, $K$ is very small)

# PCA: the inverse transform from $y_n$ to $\tilde{x}_n$

- $x_n \in \mathcal{R}^M$, $y_n \in \mathcal{R}^K$, $x_n$ and $y_n$ are two feature vectors of the same object
- $y_n = [y_{n,1} \quad \cdots \quad y_{n,K}]^T$
- Inverse transform from $y_n$ to $\tilde{x}_n$ : $\tilde{x}_n = By_n + \mu$

$$B = [w_1, \ldots, w_K] \begin{bmatrix} \lambda_1^{\frac{1}{2}} & & \\ & \ldots & \\ & & \lambda_K^{\frac{1}{2}} \end{bmatrix}$$

- $x_n \approx \tilde{x}_n = y_{n,1}\sqrt{\lambda_1}w_1 + y_{n,2}\sqrt{\lambda_2}w_2 \ldots + y_{n,K}\sqrt{\lambda_K}w_K + \mu$

# PCA: the inverse transform from $\beta_n$ to $\tilde{x}_n$

- $x_n \in \mathcal{R}^M$, $y_n \in \mathcal{R}^K$, $x_n$ and $y_n$ are two feature vectors of the same object
- Inverse transform from $y_n$ to $\tilde{x}_n$ : $\tilde{x}_n = By_n + \mu$

  where $B = \tilde{P}\sqrt{D}$, $\tilde{P} = [w_1, \ldots, w_K]$, $\sqrt{D}$=diag($\{\lambda_1^{1/2}, \ldots, \lambda_K^{1/2}\}$)

$$By_n = \underbrace{[w_1, \ldots, w_K] \begin{bmatrix} \lambda_1^{\frac{1}{2}} & & \\ & \ldots & \\ & & \lambda_K^{\frac{1}{2}} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} \frac{w_1^T}{\sqrt{\lambda_1}} \\ \vdots \\ \frac{w_K^T}{\sqrt{\lambda_K}} \end{bmatrix} (x_n - \mu)}_{y_n} = \sum_{k=1}^{K} \underbrace{w_k^T (x_n - \mu)}_{\text{a scalar } \beta_{n,k}} w_k$$

# PCA: the inverse transform from $\beta_n$ to $\tilde{x}_n$

- define the vector $\beta_n = [\beta_{n,1} \quad \dots \quad \beta_{n,K}]^T$

  where $\beta_{n,k} = w_k^T (x_n - \mu)$ is a scalar

- $x_n$ can be approximated by a linear combination of the eigenvectors, and the weights are in the vector $\beta_n$

$$x_n \approx \tilde{x}_n = \beta_{n,1} w_1 + \beta_{n,2} w_2 \dots + \beta_{n,K} w_K + \mu$$

# PCA: the inverse transform from $y_n$ or $\beta_n$ to $\tilde{x}_n$

- $x_n$ can be approximated by a linear combination of the eigenvectors:

$$x_n \approx \tilde{x}_n = \beta_{n,1} w_1 + \beta_{n,2} w_2 \ldots + \beta_{n,K} w_K + \mu$$

$$x_n \approx \tilde{x}_n = y_{n,1} \sqrt{\lambda_1} w_1 + y_{n,2} \sqrt{\lambda_2} w_2 \ldots + y_{n,K} \sqrt{\lambda_K} w_K + \mu$$

- $\beta_n = [\beta_{n,1} \quad \cdots \quad \beta_{n,K}]^T$ where $\beta_{n,k} = w_k^T (x_n - \mu)$ is a scalar
- $y_n = [y_{n,1} \quad \cdots \quad y_{n,K}]^T$ where $y_{n,k} = \beta_{n,k} / \sqrt{\lambda_k}$
- $y_n \; and \; \beta_n \in \mathcal{R}^K$ are two "new" feature vectors
- $y_n$ is the normalized feature vector $y_n = \sqrt{D^{-1}} \beta_n$ and $\beta_n = \sqrt{D}\, y_n$

$\sqrt{D}$=diag($\{\lambda_1^{1/2}, \ldots, \lambda_K^{1/2}\}$) and $\sqrt{D^{-1}}$=diag($\{\lambda_1^{-1/2}, \ldots, \lambda_K^{-1/2}\}$)

# PCA: the inverse transform from $y_n$ or $\beta_n$ to $\tilde{x}_n$

- $x_n$ can be approximated by a linear combination of the eigenvectors:

$$x_n \approx \tilde{x}_n = \tilde{P}\beta_n + \mu = \tilde{P}\sqrt{D}\ y_n + \mu$$

$$\tilde{P} = [w_1, \ldots, w_K]$$

- $\beta_n \in \mathcal{R}^K$ is a feature vector $\beta_n = \sqrt{D}\ y_n$
- $y_n \in \mathcal{R}^K$ is the normalized feature vector $y_n = \sqrt{D^{-1}}\beta_n$
- an element in $y_n$ or $\beta_n$ is a principal component of the data point $x_n$

# Explain the math of PCA transforms on one page

- Orthogonal eigenvectors: $\{w_1, \dots, w_K, K \leq M\}$, $w_k \in \mathcal{R}^M$: ${w_i}^T w_j = 0$ if $i \neq j$
- The constraint $\|w_k\|_2 = \sqrt{{w_k}^T w_k} = 1$
- Goal: to obtain $y_n = [y_{n,1} \quad \cdots \quad y_{n,K}]^T$ and $\beta_n = [\beta_{n,1} \quad \cdots \quad \beta_{n,K}]^T$
- Math:
  - $x_n = \tilde{x}_n = \beta_{n,1} w_1 + \beta_{n,2} w_2 \dots + \beta_{n,k} w_k \dots + \beta_{n,M} w_M + \mu$ (if $K = M$ )
  - $x_n - \mu = \beta_{n,1} w_1 + \beta_{n,2} w_2 \dots + \beta_{n,k} w_k + \cdots + \beta_{n,M} w_M$
  - ${w_k}^T (x_n - \mu) = \beta_{n,1} {w_k}^T w_1 + \beta_{n,2} {w_k}^T w_2 \dots + \beta_{n,k} \boldsymbol{{w_k}^T w_k} + \cdots$
  - ${w_k}^T (x_n - \mu) = \beta_{n,k}$
  - $y_{n,k} = \beta_{n,k} / \sqrt{\lambda_k}$ (by definition)

# a difference between vector notations in textbooks and Numpy array in Python

- $N$ data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$

$x_n = \begin{bmatrix} x_{n,1} \\ x_{n,2} \\ \vdots \\ x_{n,M} \end{bmatrix}$, a data point is a column vector (M by 1 matrix)

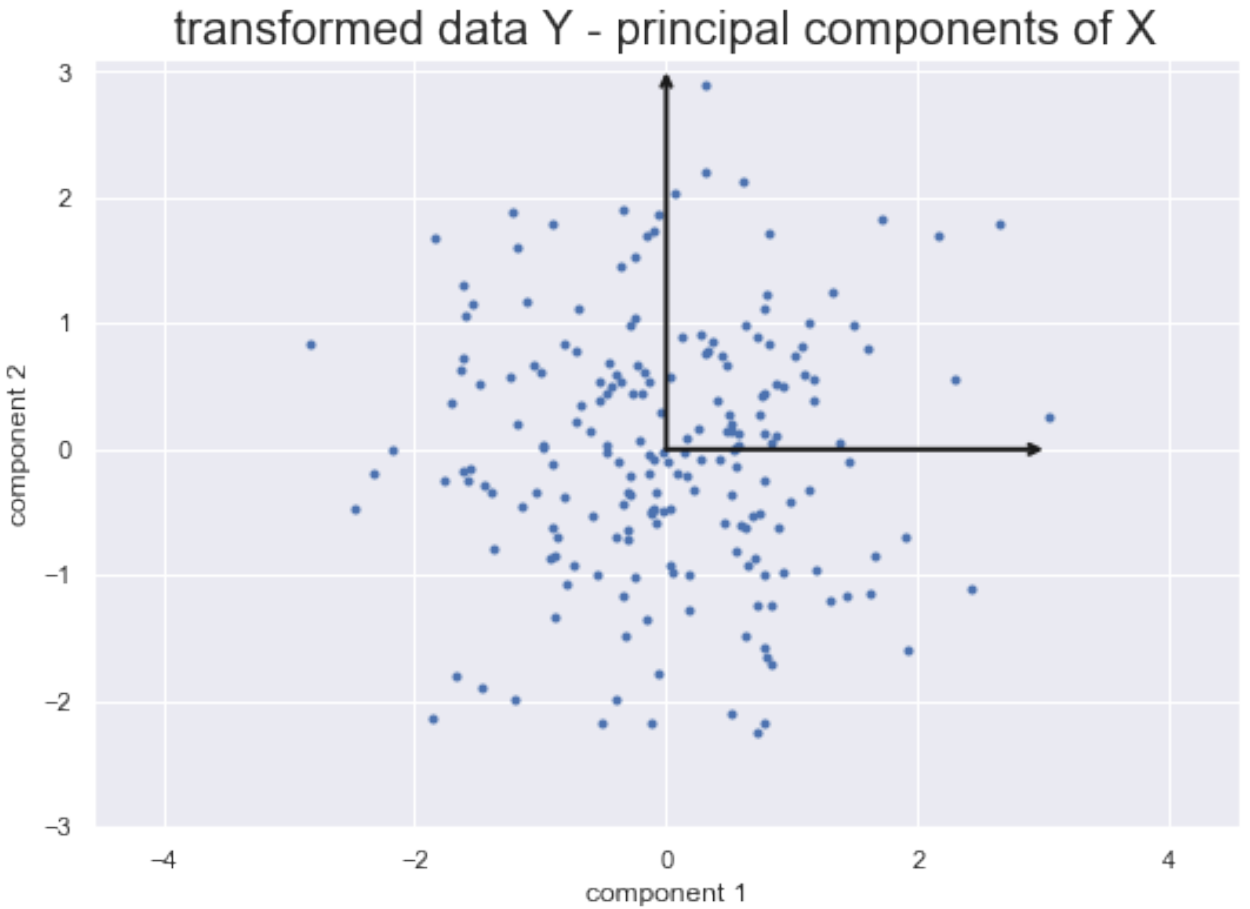
- Numpy 2D array to store data $X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}$

  Each row of the data matrix $X$ is a data point

  $X[0,:]$ is 1D array – feature vector of the object-0

see the demo PCA.ipynb

# PCA is a linear transform composed of a translation, rotation, and uniform scaling.
# It is a transform from one coordinate system to another

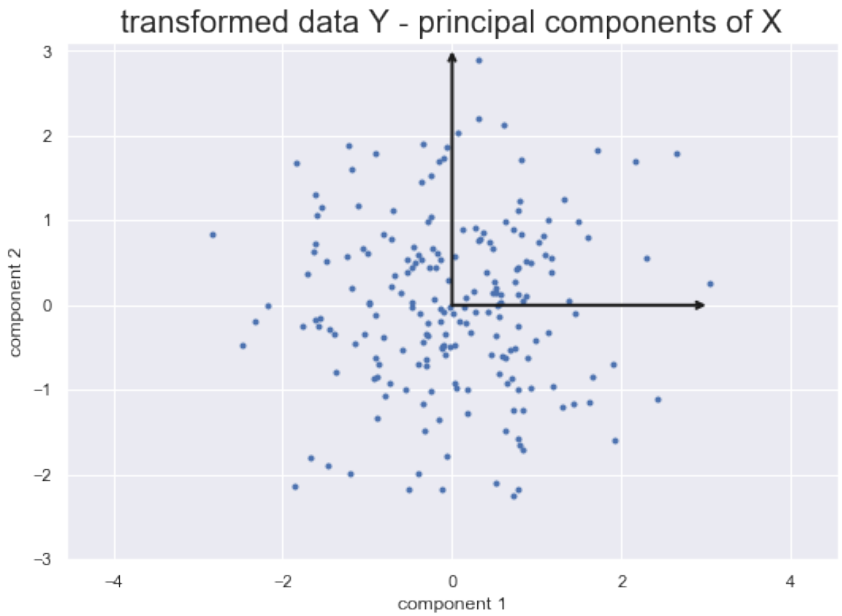

an eigenvector of $C$ is a principal axis/direction

an eigenvalue of $C$ is data variation along a principal direction

# PCA: the forward transform and the inverse transform


$$y_n = A\,(x_n - \mu)$$

forward transform (K=2)




$$\tilde{x}_n = B y_n + \mu$$

inverse transform (K=1)

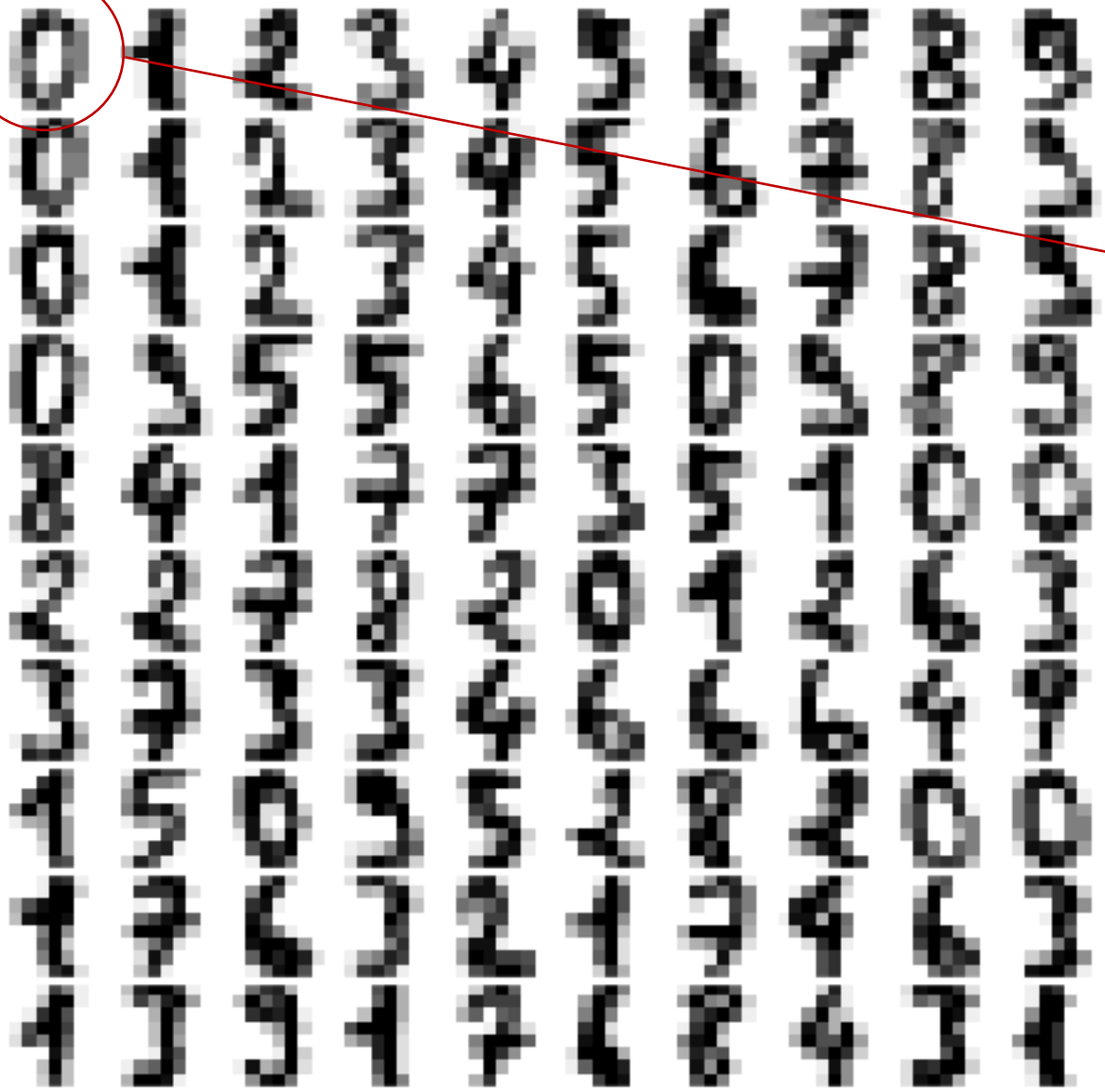
$x_n$: a blue point

$\tilde{x}_n$: a red point

$\tilde{x}_n \approx x_n$

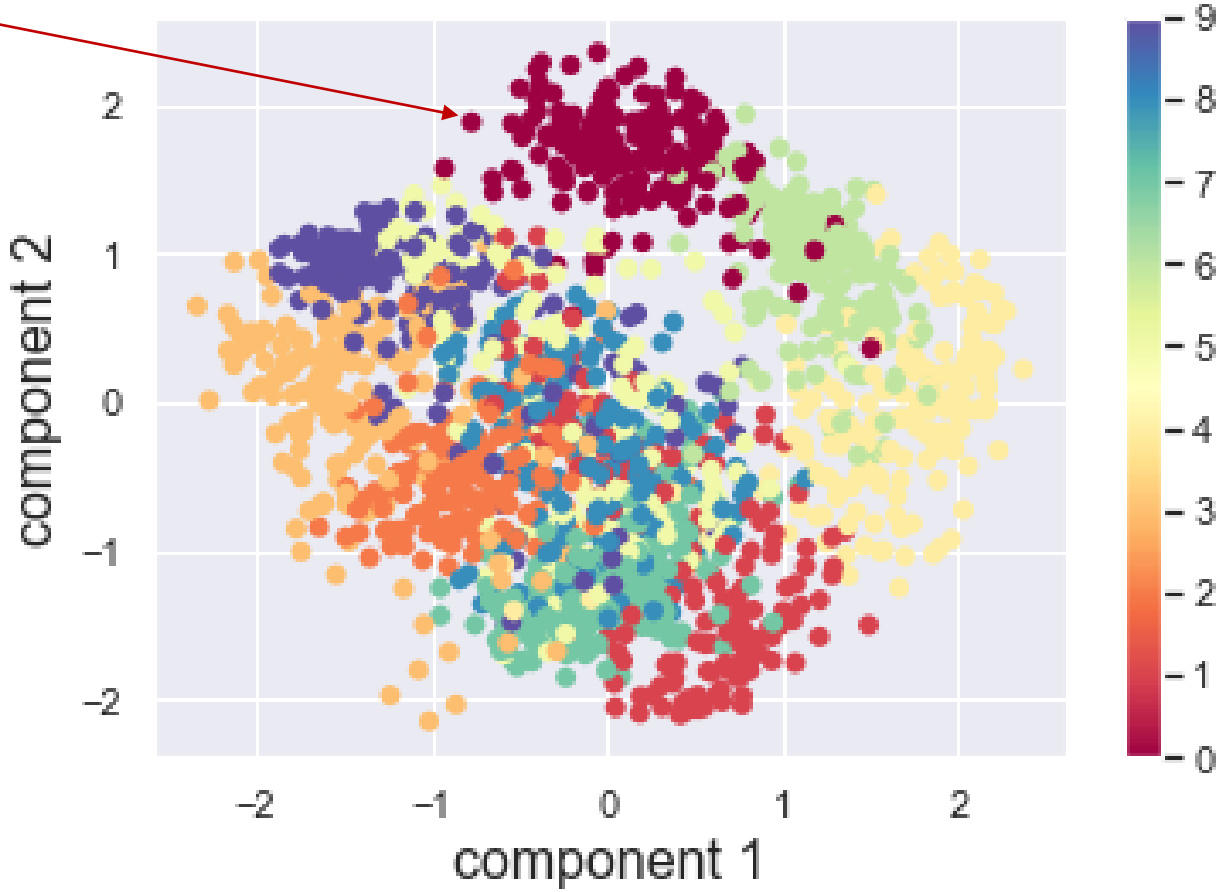
see the demo PCA_visualization_denoising.ipynb

# Use PCA to Visualize High Dimensional Data



Each 8-by-8 image has 64 pixels

$$x_n \in \mathcal{R}^{64} \Rightarrow \text{PCA} \Rightarrow y_n \in \mathcal{R}^2$$



# Use PCA for Noise Filtering

noisy image $x_n \in \mathcal{R}^{64}$

PCA
(K=12)

forward transform (K=12)

$$y_n = A\,(x_n - \mu)$$

clean image $\tilde{x}_n \in \mathcal{R}^{64}$

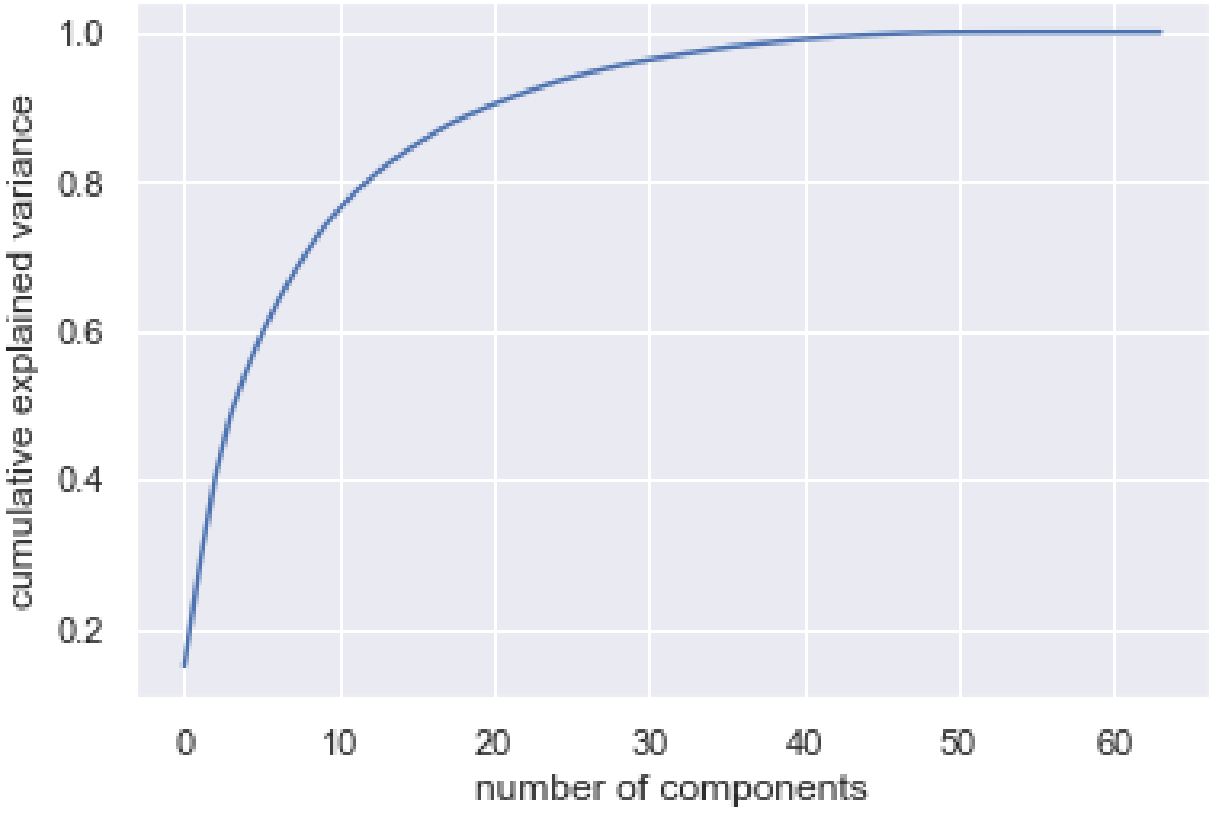
inverse transform (K=12)

$$\tilde{x}_n = By_n + \mu$$

# Choosing the number of components

$x_n \in \mathcal{R}^M, y_n \in \mathcal{R}^K$

cumulative explained variance (ratio) $r(K) = \frac{\sum_{k=1}^{K} \lambda_k}{\sum_{m=1}^{M} \lambda_m}$




$$r(1) = \frac{\lambda_1}{\sum_{m=1}^{M} \lambda_m}$$

$$r(2) = \frac{\lambda_1 + \lambda_2}{\sum_{m=1}^{M} \lambda_m}$$

$$r(K) \to 1 \; as \; K \to M$$

"the first 10 components explain/contain approximately 75% of the variance" - what does it mean ?

# Understand PCA from two perspectives

- There are two ways to get to the PCA algorithm
  - maximum variance
  - minimum reconstruction error
- Input: a set of data points $\{x_1, x_2, x_3, \dots, x_N\}$ and $x_n \in \mathcal{R}^M$
- Output:
  - a set of transformed data points $\{y_1, y_2, y_3, \dots, y_N\}$ and $y_n \in \mathcal{R}^K$, $K \leq M$, $K \ll N$
  - a set of reconstructed data points $\{\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \dots, \tilde{x}_N\}$ from the inverse transform
  - $x_n$ and $y_n$ are two feature vectors of the same object
  - $\tilde{x}_n \approx x_n$

# PCA: maximum variance

- Given a set of data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$, and the sample mean $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$

- Find a direction $w \in \mathcal{R}^M$ where $\|w\|_2 = 1$

- Such that the variance (or variation) $J(w)$ of the data along the direction $w$ is maximized:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N}\left(w^T(x_n - \mu)\right)^2$$

$w^T(x_n - \mu)$ is the scalar projection of $(x_n - \mu)$ in the direction $w$

# PCA: maximum variance

- Manipulate the objective with linear algebra

$$\left(w^T(x_n-\mu)\right)^2$$
$$=\left(w^T(x_n-\mu)\right)\left(w^T(x_n-\mu)\right)$$
$$=w^T(x_n-\mu)(x_n-\mu)^Tw$$

$$J=\frac{1}{N}\sum_{n=1}^{N}\left(w^T(x_n-\mu)\right)^2$$
$$=\frac{1}{N}\sum_{n=1}^{N}w^T(x_n-\mu)(x_n-\mu)^Tw$$
$$=w^T\left(\underbrace{\frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)(x_n-\mu)^T}\right)w$$

$$J=w^TCw$$

Covariance Matrix $C$ of the data points

# PCA: maximum variance

- Theorem: $\max_{\|w\|_2=1} w^T C w \Leftrightarrow Cw = \lambda w$

which means the solution $w$ is an eigenvector of $C$

Proof:

from Lagrangian function of the optimization problem $\max_{\|w\|_2=1} w^T C w$

$$L(w, \lambda) = w^T C w + \lambda(1 - \|w\|_2^2)$$

Get the optimal $w$ by :

$$\frac{\partial L}{\partial w} = 2Cw - 2\lambda w = 0$$

Thus:

$Cw = \lambda w$ means $w$ is an eigenvector and $\lambda$ is an eigenvalue of $C$

$J = w^T C w = w^T \lambda w = \lambda \|w\|_2^2 = \lambda$

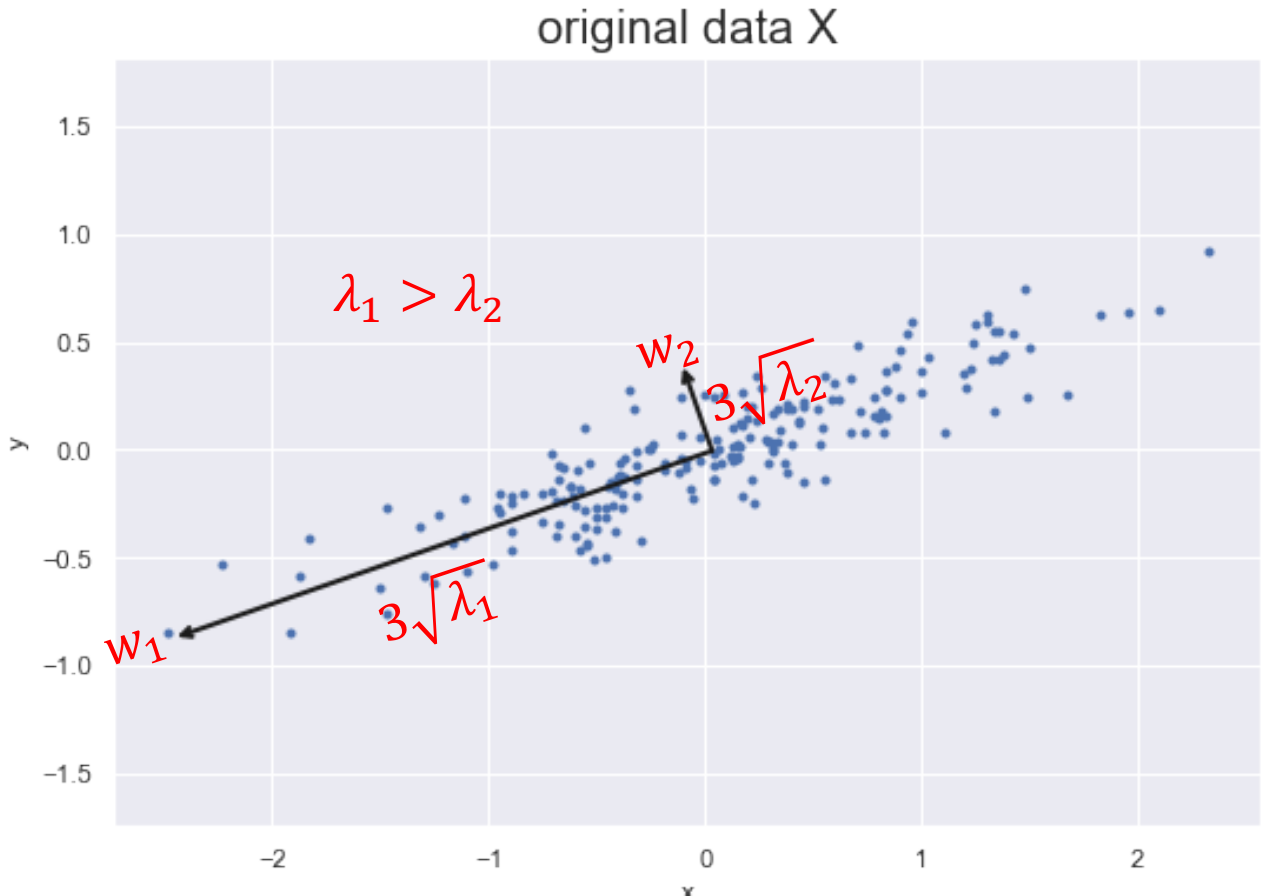
# PCA: maximum variance

- Given a set of data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$, and the sample mean $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$
- Along the direction $w \in \mathcal{R}^M$ with $Cw = \lambda w$ and $\|w\| = 1$, the variance (or variation) $J(w)$ of the data is:

$$J(w) = \frac{1}{N}\sum_{n=1}^{N}\left(w^T(x_n - \mu)\right)^2 = \lambda$$

- $\lambda$ is an eigenvalue of $C$
- $w$ is an eigenvector of $C$

$w^T(x_n - \mu)$ is the scalar projection of $(x_n - \mu)$ in the direction $w$



# PCA: minimum reconstruction error

- Given a set of data points $\{x_1, x_2, x_3, \ldots, x_N\}$ and $x_n \in \mathcal{R}^M$, and the sample mean $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$
- Find a set of vectors $\{w_1, \ldots, w_K\}$, $w_n \in \mathcal{R}^M$

  $w_i{}^T w_j = 0$ if $i \neq j$, and $\|w_i\|_2 = 1$
- Such that $x_n$ can be well approximated by a linear combination:

$$x_n \approx \tilde{x}_n = \beta_{n,1} w_1 + \beta_{n,2} w_2 \ldots + \beta_{n,K} w_K + \mu$$

- The goal is to minimize the reconstruction error

$$J(w_1, \ldots, w_K, \beta_{n,1}, \ldots, \beta_{n,K}) = \frac{1}{N}\sum_{n=1}^{N} \|x_n - \tilde{x}_n\|_2^2$$

# PCA: minimum reconstruction error

- $x_n \approx \tilde{x}_n = \beta_{n,1} w_1 + \beta_{n,2} w_2 \ldots + \beta_{n,K} w_K + \mu$
- the reconstruction error of PCA is

$$J = \frac{1}{N} \sum_{n=1}^{N} \|x_n - \tilde{x}_n\|_2^2 = \lambda_{K+1} + \lambda_{K+2} + \cdots + \lambda_M$$

$$J = 0 \text{ if } K = M$$

$w_i^T w_j = 0$ if $i \neq j$, $\|w_i\|_2 = 1$

$\beta_{n,i} = w_i^T (x_n - \mu)$

$\{w_1, \ldots, w_M\}$ are eigenvectors of $C$

$\{\lambda_1, \ldots \lambda_M\}$ are eigenvalues of $C$

# PCA: minimum reconstruction error

- the reconstruction error $J = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|_2^2 = \lambda_{K+1} + \lambda_{K+2} + \cdots + \lambda_M$

$x_n \approx \tilde{x}_n = \beta_{n,1}w_1 + \beta_{n,2}w_2 \ldots + \beta_{n,K}w_K + \mu, \quad {w_i}^T w_j = 0 \text{ if } i \neq j, \|w_i\|_2 = 1$

**Verification**: (not a proof) assume we find $M$ orthonormal basis vectors $\{w_1, \ldots, w_M\}$ which are eigenvectors of $C$, then any vector can be expressed as a linear combination of basis vectors

$x_n - \mu = (\beta_{n,1}w_1 + \beta_{n,2}w_2 \ldots + \beta_{n,K}w_K) + \beta_{n,K+1}w_{K+1} \ldots + \beta_{n,M}w_M$

${w_i}^T(x_n - \mu) = {w_i}^T w_1\beta_{n,1} + \cdots + {w_i}^T w_i\beta_{n,i} + \cdots + {w_k}^T w_K\beta_{n,K} = \beta_{n,i}$

$x_n - \tilde{x}_n = \beta_{n,K+1}w_{K+1} + \beta_{n,K+2}w_{k+2} + \cdots + \beta_{n,M}w_M$

$\|x_n - \tilde{x}_n\|_2^2 = \beta_{n,K+1}^2 + \cdots + \beta_{n,M}^2 = \sum_{i=K+1}^{M} {w_i}^T(x_n - \mu)(x_n - \mu)^T w_i$

$J = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|_2^2 = \sum_{i=K+1}^{M} {w_i}^T \underbrace{\left(\frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)(x_n - \mu)^T\right)}_{\text{covariance matrix } C} w_i = \sum_{i=K+1}^{M}\lambda_i$

a proof: machine learning a probabilistic perspective

# Explain PCA from the perspective of reconstruction error

Input: $N$ data points $\{x_1, x_2, x_3, \dots, x_N\}$ and $x_n \in \mathcal{R}^M$

Step-1: Compute the mean $\mu$ and the covariance matrix $C$ from the data

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n, \qquad C = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)\,(x_n - \mu)^T$$

Step-2: Get eigenvectors $w_1, w_2, \dots, w_K, w_{K+1}, w_{K+2}, \dots, w_M$ of $C$, and the corresponding eigenvalues: $\lambda_1 \geq \lambda_2 \dots \geq \lambda_K \geq \lambda_{K+1} \geq \lambda_{K+2} \geq \cdots \geq \lambda_M \geq 0$

Reconstruction error $J = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|_2^2 = \lambda_{K+1} + \lambda_{K+2} + \cdots + \lambda_M$

$$x_n = \mu + \beta_{n,1}w_1 + \beta_{n,2}w_2 \dots + \beta_{n,k}w_K + \beta_{n,k}w_{K+1} + \beta_{n,k}w_{K+2} \dots + \beta_{n,M}w_M$$

$\tilde{x}_n = PCA^{-1}(x_n)$ inverse transform

**ignored by PCA**

"most" information for reconstruction

"less" information

# PCA: minimum reconstruction error

- the reconstruction error $J = \frac{1}{N}\sum_{n=1}^{N}\|x_n - \tilde{x}_n\|_2^2 = \lambda_{K+1} + \lambda_{K+2} + \cdots + \lambda_M$

$x_n \approx \tilde{x}_n = \beta_{n,1}w_1 + \beta_{n,2}w_2 \ldots + \beta_{n,K}w_K + \mu$, ${w_i}^T w_j = 0$ if $i \neq j$, $\|w_i\|_2 = 1$

$\beta_{n,i} = {w_i}^T(x_n - \mu)$

$\{w_1 \ldots w_K\}$ are eigenvectors of $C$

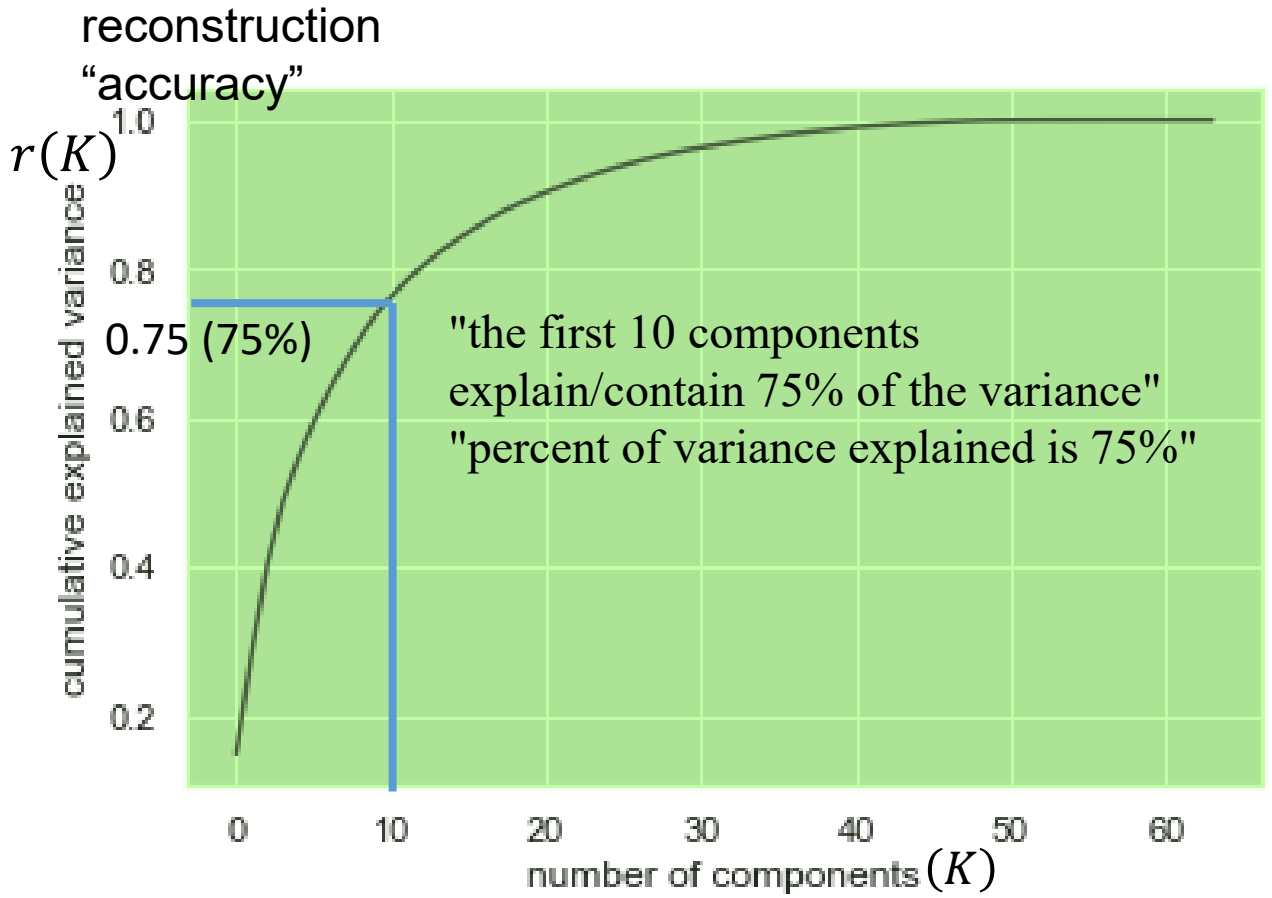
$\{\lambda_1 \ldots \lambda_K \ldots \lambda_M\}$ are eigenvalues of $C$

$\lambda_1 \geq \lambda_2 \ldots \geq \lambda_K \geq \lambda_{K+1} \geq \cdots \geq \lambda_M$

total variance= $\sum_{m=1}^{M}\lambda_m$

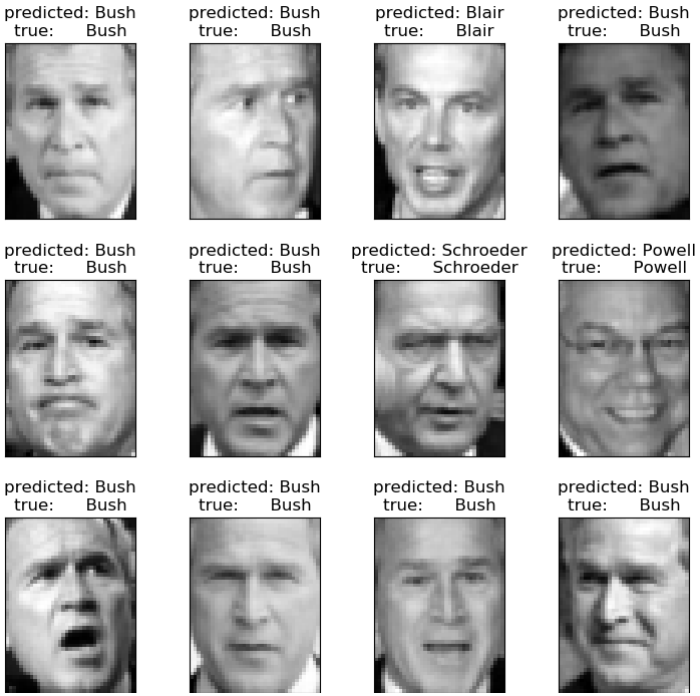
cumulative explained variance ratio:

$$r(K) = \frac{\sum_{k=1}^{K}\lambda_k}{\sum_{m=1}^{M}\lambda_m} = 1 - \frac{J}{\sum_{m=1}^{M}\lambda_m}$$



# Use PCA to Obtain Eigenfaces

image $x_n \in \mathcal{R}^{2914}$




mean face

$\mu$

The eigenfaces are eigenvectors $\{w_k\}$ of the covariance matrix $C$ of the data

Encoding: $\beta_{n,i} = w_i{}^T(x_n - \mu)$

Decoding: $x_n \approx \beta_{n,1}w_1 + \beta_{n,2}w_2 \ldots + \beta_{n,K}w_K + \mu$

# PCA is a generative model

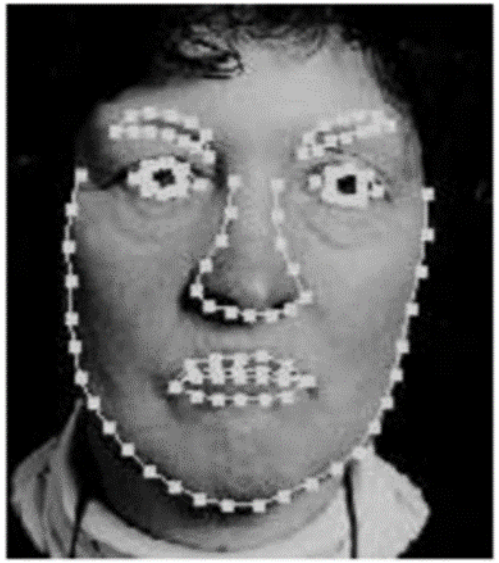
$$\tilde{x}_n = y_{n,1}\sqrt{\lambda_1}w_1 + y_{n,2}\sqrt{\lambda_2}w_2 \ldots + y_{n,K}\sqrt{\lambda_K}w_K + \mu$$

Given the eigenvectors and eigenvalues, we assign some random values to $y_{n,1}, \ldots, y_{n,K}$ to generate new images

see the demo: PCA_eigenface.ipynb

# PCA for Image Analysis
# - Statistical Shape Model (Active Shape Model)

https://www.cs.cmu.edu/~efros/courses/AP06/Papers/cootes-eccv-98.pdf



$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{b}_s$$

**Fig. 1.** Example of face image labelled with 122 landmark points

# PCA for Image Analysis -Active Appearance Model

**Fig. 2.** First two modes of shape variation (±3 sd)

**Fig. 3.** First two modes of grey-level variation (±3 sd)

much better than eigenface

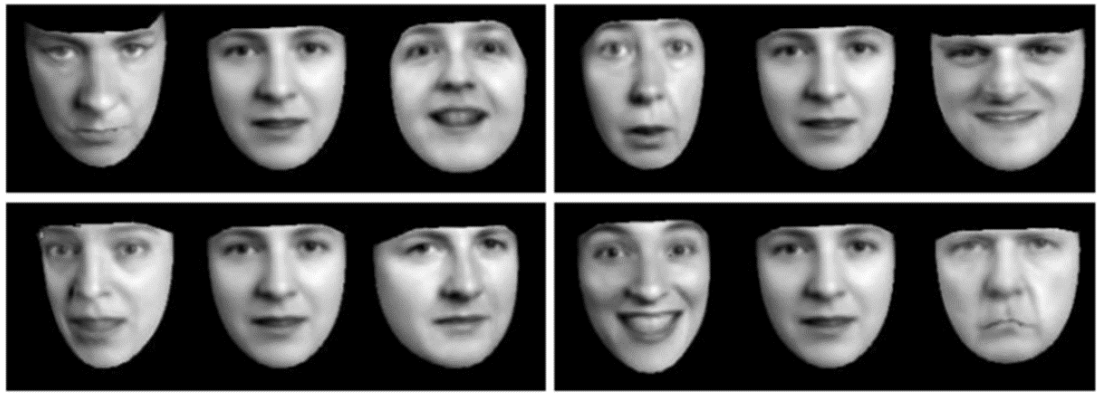


**Fig. 4.** First four modes of appearance variation (±3 sd)

# The Connection Between PCA and Multivariate Gaussian Distribution

- $f_X(x|\mu,\Sigma) = \frac{1}{(2\pi)^{M/2}|\Sigma|^{1/2}} exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$

  $x$ is a random vector, and it has $M$ elements. *exp* denotes exponential function

  $\mu = E[X]$, and $\Sigma = Cov(X,X) = E\left[(X-\mu)(X-\mu)^T\right]$

- If the data points $\{x_1, \ldots, x_N\}$ are generated from $f_X(x|\mu,\Sigma)$, then:
  - $\Sigma \approx C = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)\,(x_n-\mu)^T$
  - $\mu \approx \frac{1}{N}\sum_{n=1}^{N} x_n$
  - $x_n \in \mathcal{R}^M$
- PCA works very well if data distribution is close to a multivariate Gaussian

# PCA: the inverse transform from $y_n$ to $\tilde{x}_n$

- $x_n \in \mathcal{R}^M$, $y_n \in \mathcal{R}^K$, $x_n$ and $y_n$ are two feature vectors of the same object

$$y_n = A\,(x_n - \mu),\ A = \begin{bmatrix} w_1^T/\sqrt{\lambda_1} \\ w_2^T/\sqrt{\lambda_2} \\ \dots \\ w_K^T/\sqrt{\lambda_K} \end{bmatrix} = \sqrt{D^{-1}}\tilde{P}^T,\ A \in \mathcal{R}^{K\times M},\ \text{and } {w_i}^T w_j = 0 \text{ if } i \neq j$$

$$\tilde{P} = [w_1, \dots, w_K],\ \sqrt{D^{-1}} = \text{diag}(\{\lambda_1^{-1/2}, \dots, \lambda_K^{-1/2}\})$$

- Inverse transform from $y_n$ to $\tilde{x}_n$

$\tilde{x}_n = By_n + \mu$, where $B = \tilde{P}\sqrt{D}$ and $\sqrt{D} = \text{diag}(\{\lambda_1^{1/2}, \dots, \lambda_K^{1/2}\})$

note: $BA = \tilde{P}\sqrt{D}\sqrt{D^{-1}}\tilde{P}^T = \tilde{P}\tilde{P}^T = Q$, which is a $M \times M$ identity matrix only if $K = M$

Thus, $\tilde{x}_n = BA\,(x_n - \mu) + \mu = Qx_n + (I - Q)\mu$

if $K = M$, then $Q = I \Rightarrow \tilde{x}_n = x_n$, otherwise $\tilde{x}_n \approx x_n$