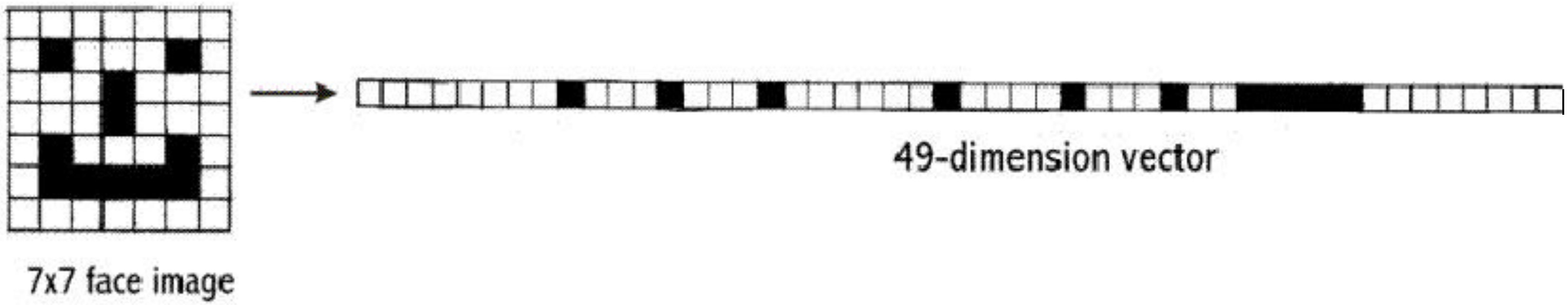
Image feature representation:

Dimensionality reduction

High Dimensional Correlated Data

- Images as a high dimensional vector



- A typical image used for image processing will be 512x512= 262144 dimension vector!
- (Registered) face images are highly correlated

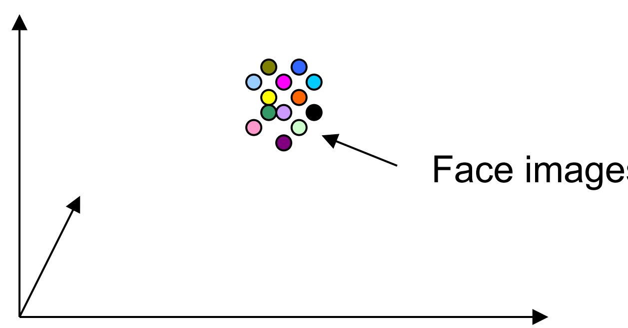


Image space (high dimensional space of all possible images)

Starting idea of "eigenfaces"

1. Treat pixels as a vector

$$\mathbf{x}$$

2. Recognize face by nearest neighbor

$$\mathbf{y}_1 \ldots \mathbf{y}_n$$

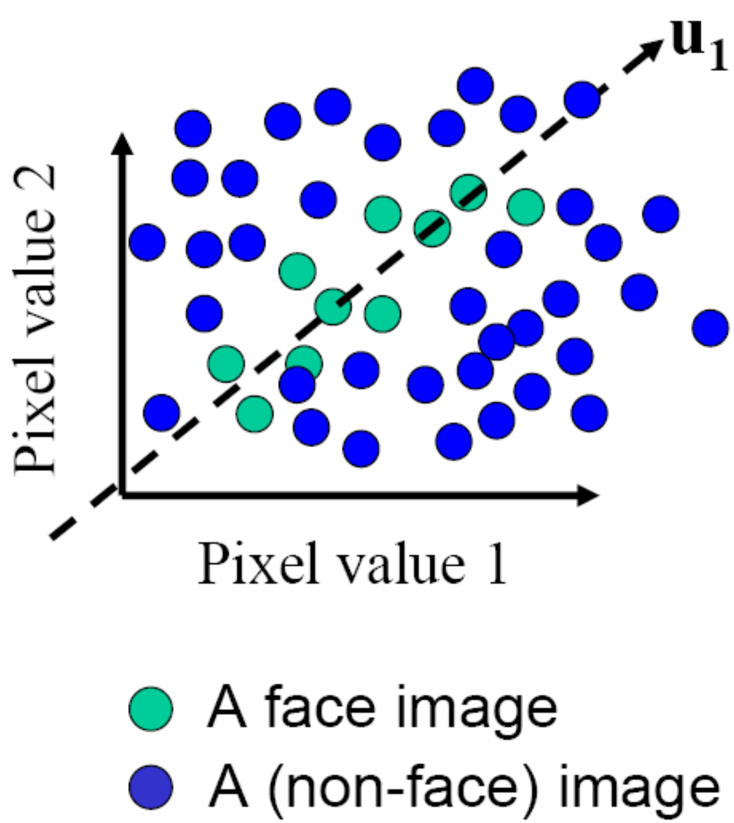
$$k = \underset{k}{\operatorname{argmin}} \left\| \mathbf{y}_k{}^T - \mathbf{x} \right\|$$

The space of all face images

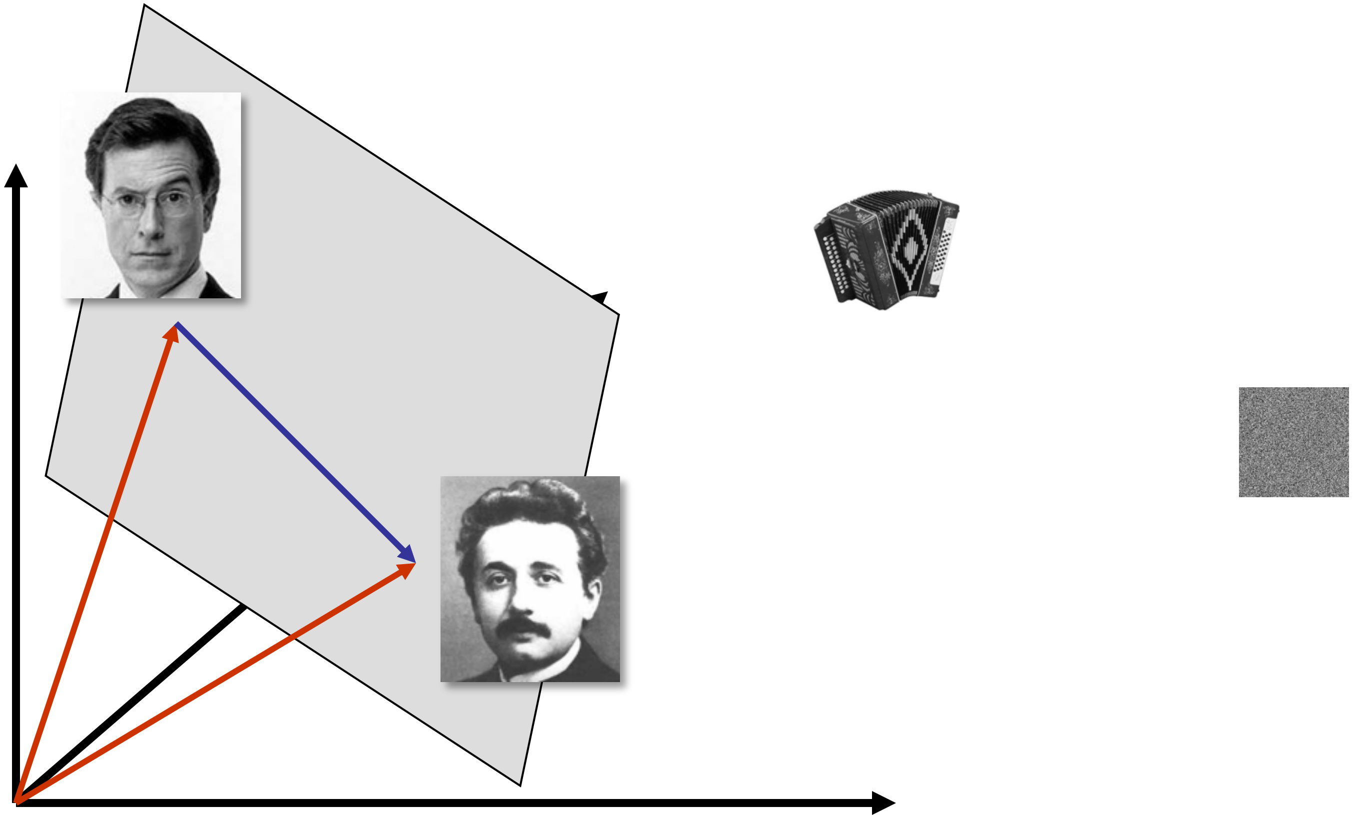
- When viewed as vectors of pixel values, face images are extremely high-dimensional
  – 512x512 image = 262,144 dimensions
  – Slow and lots of storage
- But very few 262,144D long vectors are valid face images; real face vectors are sparse (i.e. sparse distribution) in the data space.
- We want to effectively model the subspace of face images

The space of all face images

- Eigenface idea:
  - Construct a low-dimensional linear subspace that best explains the variation in the set of face images



Dimensionality reduction



- The set of faces is a "subspace" of the set of all images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a "hyper-plane" to the set of faces
    - spanned by vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_K}$
    - any face $\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

PCA

- General dimensionality reduction technique
- Preserves most of variance with a much more compact representation
  - Lower storage requirements (eigenvectors + a few numbers per face)
  - Faster matching

Principal Component Analysis (PCA)

- Given: N data points $\mathbf{x_1}, \ldots, \mathbf{x_N}$ in R^d
- We want to find a new set of features that are linear combinations of original ones:

$$w = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$

(**μ**: mean of data points)

- Choose unit vector **u** in R^d that captures the most data variance

Principal Component Analysis

- Direction that maximizes the variance of the projected data:

Maximize $$\frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu)}_{\text{Projection of data point}}(\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu))^{\mathrm{T}}$$ subject to ||**u**||=1

$$= \quad \mathbf{u}^{\mathrm{T}}\underbrace{\left[1/N\sum_{i=1}^{N}(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^{\mathrm{T}}\right]}_{\text{Covariance matrix of data}}\mathbf{u}$$

$$= \quad \mathbf{u}^{\mathrm{T}}\Sigma\mathbf{u}$$

→ The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ

Another perspective

- Minimize approximation error

$$
\begin{aligned}
\|\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}\|^2 &= \left(\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}\right)\cdot\left(\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}\right) \\
&= \vec{x_i}\cdot\vec{x_i} - \vec{x_i}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} \\
&\quad -(\vec{w}\cdot\vec{x_i})\vec{w}\cdot\vec{x_i} + (\vec{w}\cdot\vec{x_i})\vec{w}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} \\
&= \|\vec{x_i}\|^2 - 2(\vec{w}\cdot\vec{x_i})^2 + (\vec{w}\cdot\vec{x_i})^2\vec{w}\cdot\vec{w} \\
&= \vec{x_i}\cdot\vec{x_i} - (\vec{w}\cdot\vec{x_i})^2
\end{aligned}
$$

Another perspective

- Minimize approximation error

$$
\begin{aligned}
MSE(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n} \|\vec{x_i}\|^2 - (\vec{w}\cdot\vec{x_i})^2 \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} \|\vec{x_i}\|^2 - \sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2\right)
\end{aligned}
$$

Another perspective

- Minimize approximation error

$$
\begin{aligned}
MSE(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n} \|\vec{x_i}\|^2 - (\vec{w}\cdot\vec{x_i})^2 \\
&= \frac{1}{n}\left(\sum_{i=1}^{n} \|\vec{x_i}\|^2 - \sum_{i=1}^{n} (\vec{w}\cdot\vec{x_i})^2\right)
\end{aligned}
$$

Equivalent to Maximize: $\sum_{i=1}^{n} (\vec{w}\cdot\vec{x_i})^2$

Another perspective

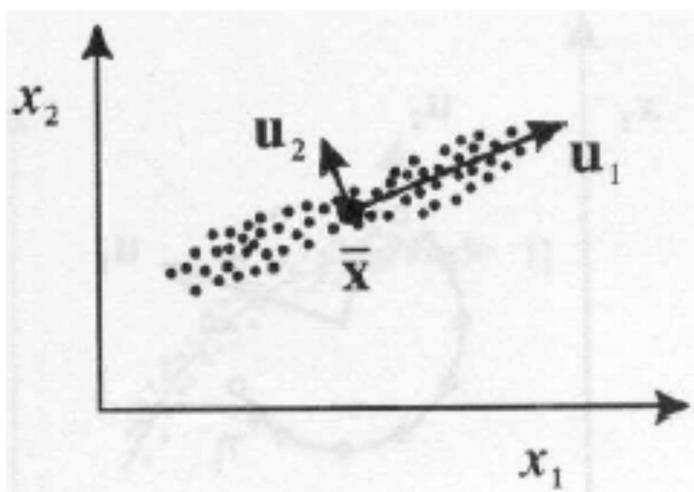
- Maximize variance

$$
\begin{aligned}
\sigma_{\vec{w}}^2 &= \frac{1}{n}\sum_i \left(\vec{x_i}\cdot\vec{w}\right)^2 \\
&= \frac{1}{n}(\mathbf{xw})^T(\mathbf{xw}) \\
&= \frac{1}{n}\mathbf{w}^T\mathbf{x}^T\mathbf{xw} \\
&= \mathbf{w}^T\frac{\mathbf{x}^T\mathbf{x}}{n}\mathbf{w} \\
&= \mathbf{w}^T\mathbf{vw}
\end{aligned}
$$

Please refer to the attached document for details derivations.

Geometric interpretation

- PCA projects the data along the directions where the data varies most.
- These directions are determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues.
- The magnitude of the eigenvalues corresponds to the variance of the data along the eigenvector directions.



PCA for dimension reduction

- Lower dimensionality basis
  - Approximate vectors by finding a basis in an appropriate lower dimensional space.

(1) Higher-dimensional space representation:

$$x = a_1 v_1 + a_2 v_2 + \cdots + a_N v_N$$

$v_1, v_2, ..., v_N$ is a basis of the N-dimensional space

(2) Lower-dimensional space representation:

$$\hat{x} = b_1 u_1 + b_2 u_2 + \cdots + b_K u_K$$

$u_1, u_2, ..., u_K$ is a basis of the K-dimensional space

- *Note:* if both bases have the same size ($N = K$), then $x = \hat{x}$)

PCA Algorithm

– Suppose $x_1, x_2, \ldots, x_M$ are N x 1 vectors

Step 1: $\bar{x} = \frac{1}{M} \sum_{i=1}^{M} x_i$

Step 2: subtract the mean: $\Phi_i = x_i - \bar{x}$ (i.e., center at zero)

Step 3: form the matrix $A = [\Phi_1 \ \Phi_2 \cdots \Phi_M]$ (NxM matrix), then compute:

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = \frac{1}{M} A A^T$$

(sample **covariance** matrix, NxN, characterizes the *scatter* of the data)

Step 4: compute the eigenvalues of C: $\lambda_1 > \lambda_2 > \cdots > \lambda_N$

Step 5: compute the eigenvectors of C: $u_1, u_2, \ldots, u_N$

PCA Algorithm

- Since C is symmetric, $u_1, u_2, \ldots, u_N$ form a basis, (i.e., any vector x or actually $(x - \bar{x})$, can be written as a linear combination of the eigenvectors):

$$x - \bar{x} = b_1 u_1 + b_2 u_2 + \cdots + b_N u_N = \sum_{i=1}^{N} b_i u_i \qquad b_i = \frac{(x - \bar{x}).u_i}{(u_i.u_i)}$$

Step 6: (**dimensionality reduction step**) keep only the terms corresponding to the K largest eigenvalues:

$$\hat{x} - \bar{x} = \sum_{i=1}^{K} b_i u_i \text{ where } K << N$$

- The representation of $\hat{x} - \bar{x}$ into the basis $u_1, u_2, \ldots, u_K$ is thus

$$\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_K \end{bmatrix}$$

PCA algorithm

- The linear transformation $R^N \rightarrow R^K$ that performs the dimensionality reduction is:

$$\begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_K \end{bmatrix} = \begin{bmatrix} u_1^T \\ u_2^T \\ \cdots \\ u_K^T \end{bmatrix} (x - \bar{x}) = U^T (x - \bar{x})$$

(i.e., simply computing coefficients of linear expansion)

Eigenfaces (PCA on face images)

1. Compute covariance matrix of face images

2. Compute the principal components ("eigenfaces")
   – K eigenvectors with largest eigenvalues

3. Represent all face images in the dataset as linear combinations of eigenfaces
   – Perform nearest neighbor on these coefficients

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

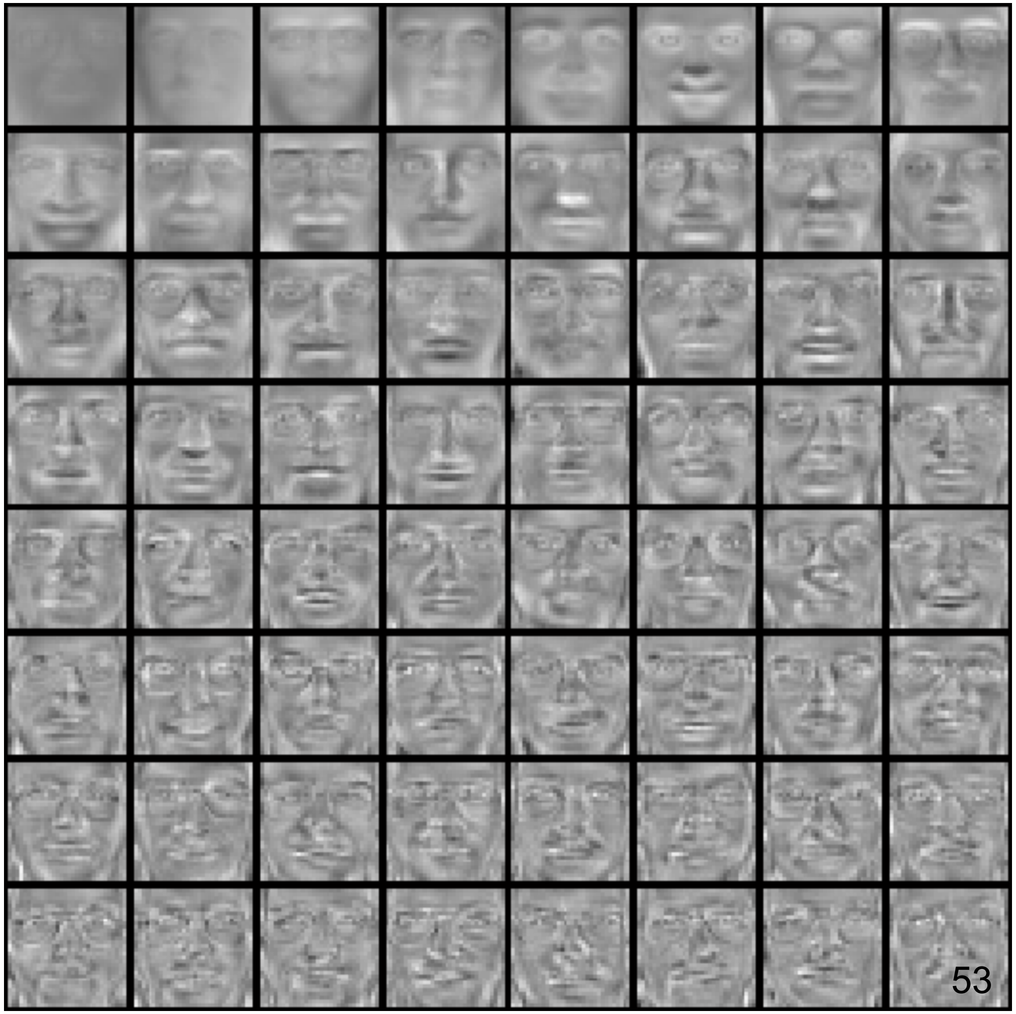
Eigenfaces example

- Training images
- $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Eigenfaces example

Mean: μ

Top eigenvectors: $u_1, \ldots u_k$



Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

Representation and reconstruction

- Face **x** in “face space” coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

 =  + 

$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

Reconstruction example

- The visualization of eigenvectors:

These are the first 4 eigenvectors from a training set of 400 images (ORL Face Database). They look like faces, hence called Eigenface.

Reconstruction

After computing eigenfaces using 400 face images from ORL face database

How to choose K?

- Choose *K* using the following criterion:

$$\frac{\sum_{i=1}^{K} \lambda_i}{\sum_{i=1}^{N} \lambda_i} > Threshold \quad \text{(e.g., 0.9 or 0.95)}$$

- In this case, we say that we "preserve" 90% or 95% of the information (variance) in the data.

- If K=N, then we "preserve" 100% of the information in the data.

Eigenfaces

Computed using 400 face images from ORL face database

Recognition with eigenfaces

Process labeled training images

- Find mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Find k principal components (eigenvectors of $\boldsymbol{\Sigma}$) $\mathbf{u}_1,\ldots\mathbf{u}_k$
- Project each training image $\mathbf{x}_i$ onto subspace spanned by principal components:
  $(w_{i1},\ldots,w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \ldots , \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

Given novel image $\mathbf{x}$

- Project onto subspace:
  $(w_1,\ldots,w_k) = (\mathbf{u}_1^T(\mathbf{x} - \boldsymbol{\mu}), \ldots , \mathbf{u}_k^T(\mathbf{x} - \boldsymbol{\mu}))$
- Optional: check reconstruction error $\hat{\mathbf{x}} - \mathbf{x}$ to determine whether image is really a face
- Classify as closest training face in k-dimensional subspace

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

Reconstruction from partial information

- Robust to partial face occlusion.

Input Reconstructed

Limitations

Global appearance method: not robust to misalignment, background variation

Limitations

- The direction of maximum variance is not always good for classification