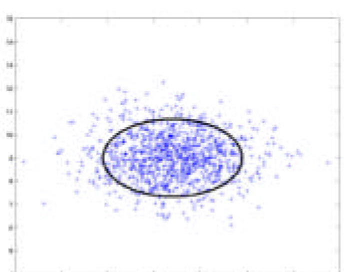
Principal Component Analysis (PCA)

Tim Marks, Cognitive Science Department

The Gaussian in *D* dimensions

- What does a set of equiprobable points look like for a 2D Gaussian?

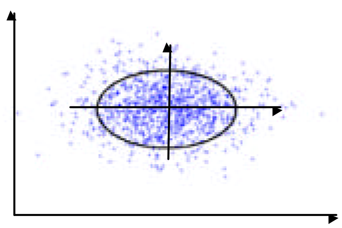


- In 2D, it's an ellipse.
- In *D* dimensions, it's an ellipsoid.

Tim Marks, Cognitive Science Department

Equiprobable contours of a Gaussian

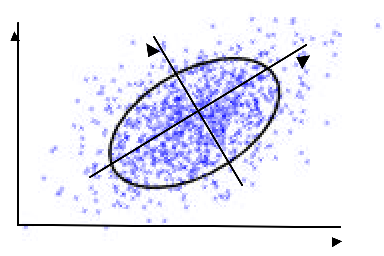
- If a Gaussian random vector has covariance matrix that is diagonal (all of the variables are uncorrelated)
  - Then the axes of the ellipsoid are parallel to the coordinate axes.



UCSD Tim Marks, Cognitive Science Department

Equiprobable contours of a Gaussian

- If a Gaussian random vector has covariance matrix that is not diagonal (some of the variables are correlated)
  - Then the axes of the ellipsoid are perpendicular to each other, but are not parallel to the coordinate axes.

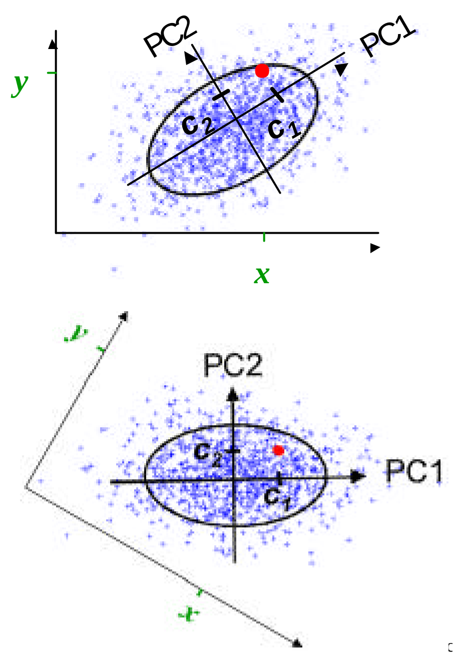


UCSD

Principle Component Analysis

- Principle component analysis (PCA) finds the directions of the axes of the ellipsoid.
- There are two ways to think about what PCA does next:
  - Projects every point perpendicularly onto the axes of the ellipsoid.
  - Rotates the ellipsoid so its axes are parallel to the coordinate axes, and translates the ellipsoid so its center is at the origin.

Tim Marks, Cognitive Science Department



Two views of what PCA does

- Transforms $\begin{array}{ccc} \mathbf{x} & & \mathbf{c} \\ \begin{bmatrix} x \\ y \end{bmatrix} & \to & \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \end{array}$
  - Projects every point perpendicularly onto the axes of the ellipsoid.
  - Rotates space so that the ellipsoid lines up with the coordinate axes, and translates space so the ellipsoid's center is at the origin.

Transformation matrix for PCA

- Let V be the matrix whose columns are the eigenvectors of the covariance matrix, S.
  - The eigenvectors $\mathbf{v}_i$ are all normalized to have length 1

$$\begin{array}{c} V \\ \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \end{array} \qquad \begin{array}{c} V^{\mathrm{T}} \\ \begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_D & \rightarrow \end{bmatrix} \end{array}$$

- The rotation transformation is given by V^{T}

UCSD Tim Marks, Cognitive Science Department

Transformation matrix for PCA

– PCA transforms the point $\boldsymbol{x}$ (original coordinates) into the point $\boldsymbol{c}$ (new coordinates).

- by subtracting the mean: $\mathbf{z} = \boldsymbol{x} - \boldsymbol{m}$
- and multiplying the result by V^{T}

$$\begin{array}{c} V^{\mathrm{T}} \\ \begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_D & \rightarrow \end{bmatrix} \end{array} \begin{array}{c} \mathbf{z} \\ \begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix} \end{array} = \begin{array}{c} \mathbf{c} \\ \begin{bmatrix} \mathbf{v}_1 \cdot \mathbf{z} \\ \mathbf{v}_2 \cdot \mathbf{z} \\ \vdots \\ \mathbf{v}_D \cdot \mathbf{z} \end{bmatrix} \end{array}$$

- Can think of as rotation because V^{T} is an orthonormal matrix
- Can think of as projection of $\mathbf{z}$ onto PC axes, because $\boldsymbol{v}_1 \bullet \mathbf{z}$ is projection of $\mathbf{z}$ onto PC1 axis, $\boldsymbol{v}_2 \bullet \mathbf{z}$ is projection onto PC2 axis, etc.

UCSD Tim Marks, Cognitive Science Department

Point is a weighted sum of eigenvectors

$$\begin{array}{ccc} V^{\mathrm{T}} & \mathbf{z} & & \mathbf{c} \\ \begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_D & \rightarrow \end{bmatrix} & \begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix} & = & \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_D \end{bmatrix} \end{array}$$

- To both sides of the equation, multiply on the left by V:

  $VV^{\mathrm{T}}\mathbf{z} = Vc.$ Because V is orthonormal, $VV^{\mathrm{T}} = I$:

  $I\mathbf{z} = Vc$

- PCA expresses the mean-subtracted point, $\mathbf{z} = \boldsymbol{x} - \boldsymbol{m}$, as a weighted sum of the eigenvectors $\mathbf{v}_i$:

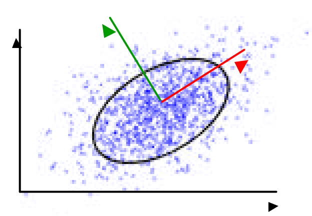
$$\begin{array}{ccccc} \mathbf{z} & = & V & \mathbf{c} & \\ \begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix} & = & \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_D \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} & \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_D \end{bmatrix} & = c_1 \begin{bmatrix} \uparrow \\ \mathbf{v}_1 \\ \downarrow \end{bmatrix} + c_2 \begin{bmatrix} \uparrow \\ \mathbf{v}_2 \\ \downarrow \end{bmatrix} + \cdots + c_D \begin{bmatrix} \uparrow \\ \mathbf{v}_D \\ \downarrow \end{bmatrix} \end{array}$$

Eigenvalues and variance

– The eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ of the covariance matrix have corresponding eigenvalues $l_1,\ l_1,\ \ldots,\ l_n$.

- It turns out that l_1 is the variance of the distribution in the $\boldsymbol{v}_1$ direction, l_2 is the variance of the distribution in the $\boldsymbol{v}_2$ direction, and so on.
  – The largest eigenvalue corresponds to the principal component in the direction of greatest variance, the next largest eigenvalue corresponds to the principal component in the perpendicular direction of next greatest variance, etc.

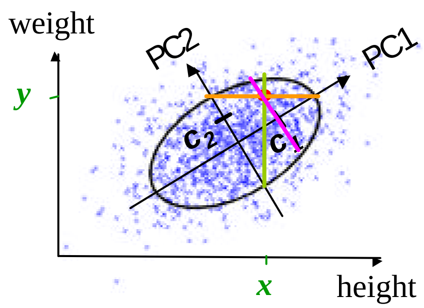
– Which eigenvector (green or red) corresponds to the smaller eigenvalue?



PCA for data compression

– What if you wanted to transmit someone's height and weight, but you could only give a single number?

- Could give only height, x
  - — = (uncertainty when height is known)
- Could give only weight, y
  - — = (uncertainty when weight is known)
- Could give only c_1, the value of first PC
  - — = (uncertainty when first PC is known)
  - – Giving the first PC minimizes the squared error of the result.



– To compress D-dimensional data into k dimensions, order the principal components in order of largest-to-smallest eigenvalue, and only save the first k components.

PCA for data compression

- Equivalent view: To compress D-dimensional data into k dimensions, order the eigenvectors in order of largest-to-smallest eigenvalue, and only use the first k eigenvectors.

$$\underset{V^{\mathrm{T}}}{\begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_k & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_D & \rightarrow \end{bmatrix}} \underset{\mathbf{z}}{\begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix}} = \underset{\mathbf{c}}{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \\ \vdots \\ c_D \end{bmatrix}} \qquad \underset{\bar{V}^{\mathrm{T}}}{\begin{bmatrix} \leftarrow & \mathbf{v}_1 & \rightarrow \\ \leftarrow & \mathbf{v}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{v}_k & \rightarrow \end{bmatrix}} \underset{\mathbf{z}}{\begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix}} = \underset{\bar{\mathbf{c}}}{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}}$$

- PCA *approximates* the mean-subtracted point, $\mathbf{z} = \boldsymbol{x} - \boldsymbol{m}$, as a weighted sum of the first k eigenvectors:

$$\underset{\bar{\mathbf{z}}}{\begin{bmatrix} \uparrow \\ \bar{\mathbf{z}} \\ \downarrow \end{bmatrix}} = \underset{\bar{V}}{\begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_k \\ \downarrow & \downarrow & & \downarrow \end{bmatrix}} \underset{\bar{\mathbf{c}}}{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}} = c_1 \begin{bmatrix} \uparrow \\ \mathbf{v}_1 \\ \downarrow \end{bmatrix} + c_2 \begin{bmatrix} \uparrow \\ \mathbf{v}_2 \\ \downarrow \end{bmatrix} + \cdots + c_k \begin{bmatrix} \uparrow \\ \mathbf{v}_k \\ \downarrow \end{bmatrix}$$

Face space

- 120×100 pixel grayscale 2D images of faces
  – Each image is considered to be a single point in face space (a single sample from a random vector): $\begin{bmatrix} \uparrow \\ x \\ \downarrow \end{bmatrix}$
- How many dimensions is the vector x ?

  $D = 120 \bullet 100 = 12000$

  – Each dimension (each pixel) can take real values between 0 and 255.
  – You can visualize in 12000 dimensions!

PCA on face images

– Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ be the n sample images.
- Find the mean image, $\boldsymbol{m}$, and subtract it from each image: $\mathbf{z}_i = \boldsymbol{x}_i - \boldsymbol{m}$
- Let A be the matrix whose columns are the mean-subtracted sample images.

$$A = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{z}_1 & \mathbf{z}_2 & \cdots & \mathbf{z}_n \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \qquad 12000 \times n \text{ matrix}$$

- Estimate the covariance matrix: $\Sigma = \mathrm{Cov}(A) = \frac{1}{n} A A^{\mathrm{T}}$

  What are the dimensions of Σ ?

The transpose trick

- The eigenvectors of Σ are the principal component directions
    - Σ is a 12000×12000 matrix
    - Too big for us to find its eigenvectors numerically
    - The first n eigenvalues are useful; the rest will all be zero.
  - Use a trick. $\Sigma = \mathrm{Cov}(A) = \frac{1}{n} A A^{\mathrm{T}}$
    - Eigenvectors of Σ are eigenvectors of AA^{T} ; still 12000×12000
    - Instead of eigenvectors of AA^{T}, we find eigenvectors of $A^{\mathrm{T}}A$
- What are the dimensions of $A^{\mathrm{T}}A$?
- $n \times n$ matrix is easy to find eigenvectors of, since n (the number of sample images) is relatively small.

UCSD Tim Marks, Cognitive Science Department

The transpose trick

- We want the eigenvectors of AA^{T}, but instead we calculated the eigenvectors of $A^{\mathrm{T}}A$. Now what?
  - Let $\mathbf{v}_i$ be an eigenvector of $A^{\mathrm{T}}A$, whose eigenvalue is l_i

$$(A^{\mathrm{T}}A)v_i = l_i v_i$$

$$A\,(A^{\mathrm{T}}Av_i) = A(l_i v_i)$$

$$AA^{\mathrm{T}}(Av_i) = l_i\,(Av_i)$$

  - Thus if $\mathbf{v}_i$ is an eigenvector of $A^{\mathrm{T}}A$, Av_i is an eigenvector of AA^{T} , with the same eigenvalue.

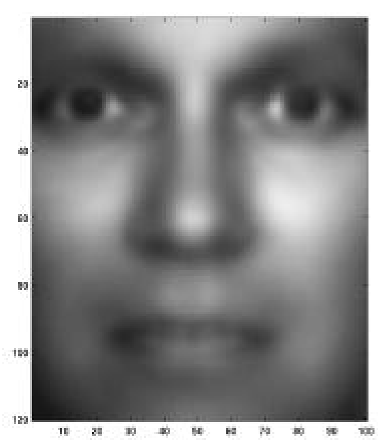
$Av_1, Av_2, \dots, Av_n$ are the eigenvectors of Σ.

$u_1 = Av_1,\ u_2 = Av_2,\ \dots,\ u_n = Av_n$ are called *eigenfaces*.

UCSD Tim Marks, Cognitive Science Department

The original images (12 out of 97)

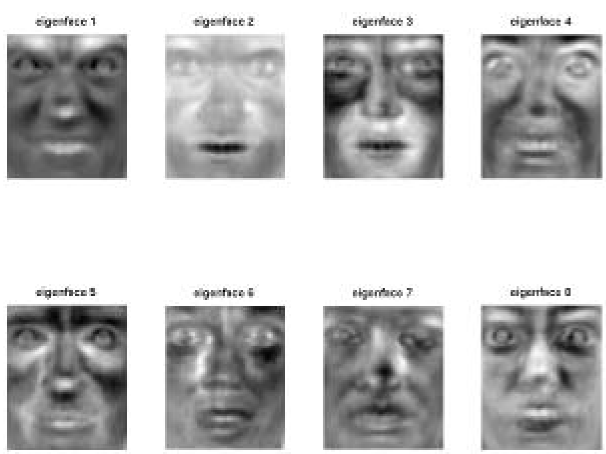
The mean face



Eigenfaces

- Eigenfaces (the principal components of face space) provide a low-dimensional representation of any face, which can be used for:
  - Face recognition
  - Facial expression recognition
  - Image reconstruction

The first 8 eigenfaces



Tim Marks, Cognitive Science Department

PCA for face representation

- To approximate a face using k dimensions, order the eigenfaces in order of largest-to-smallest eigenvalue, and only use the first k eigenfaces.

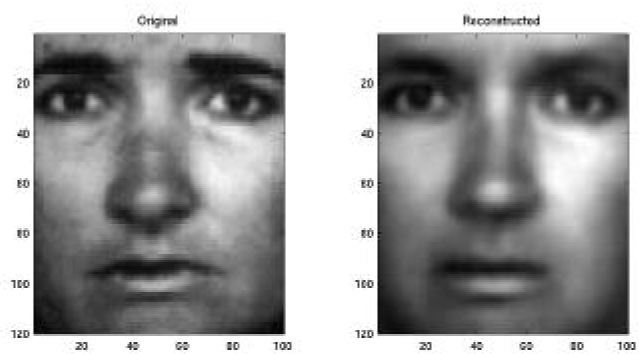
$$\begin{array}{ccccc} \bar{U}^{\mathrm{T}} & \mathbf{z} & & \bar{\mathbf{c}} \\ \begin{bmatrix} \leftarrow & \mathbf{u}_1 & \rightarrow \\ \leftarrow & \mathbf{u}_2 & \rightarrow \\ & \vdots & \\ \leftarrow & \mathbf{u}_k & \rightarrow \end{bmatrix} & \begin{bmatrix} \uparrow \\ \mathbf{z} \\ \downarrow \end{bmatrix} & = & \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} \end{array}$$

- PCA *approximates* a mean-subtracted face, $\mathbf{z} = \boldsymbol{x} - \boldsymbol{m}$, as a weighted sum of the first k eigenfaces:

$$\begin{array}{ccccc} \bar{\mathbf{z}} & = & \bar{U} & \bar{\mathbf{c}} & \\ \begin{bmatrix} \uparrow \\ \bar{\mathbf{z}} \\ \downarrow \end{bmatrix} & = & \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_k \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} & \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix} & = c_1 \begin{bmatrix} \uparrow \\ \mathbf{u}_1 \\ \downarrow \end{bmatrix} + c_2 \begin{bmatrix} \uparrow \\ \mathbf{u}_2 \\ \downarrow \end{bmatrix} + \cdots + c_k \begin{bmatrix} \uparrow \\ \mathbf{u}_k \\ \downarrow \end{bmatrix} \end{array}$$

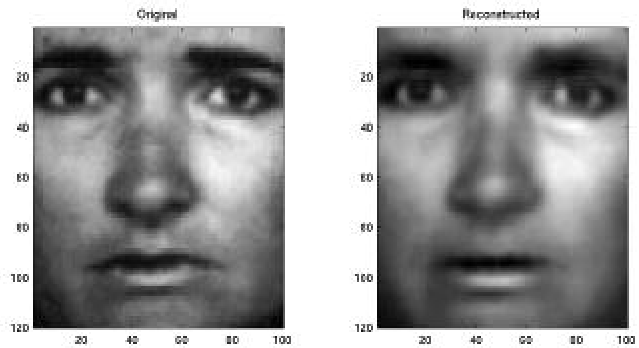
Tim Marks, Cognitive Science Department

Approximating a face using the first 10 eigenfaces

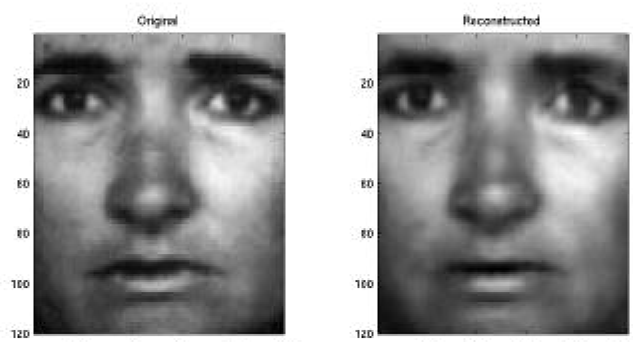


Tim Marks, Cognitive Science Department

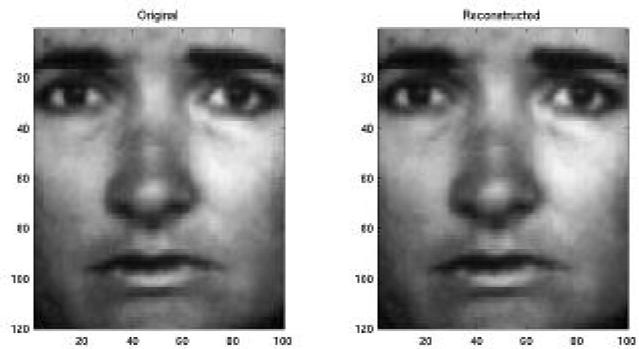
Approximating a face using the first 20 eigenfaces



Approximating a face using the first 40 eigenfaces



Approximating a face using all 97 eigenfaces



Since the image was in the original training set, the reconstruction using all 97 eigenfaces is exact. For a new face image, however, its projection into face space will only approximately match the original.

UCSD Tim Marks, Cognitive Science Department

What regularities does each eigenface capture?

- <Matlab movie>

UCSD Tim Marks, Cognitive Science Department

Eigenfaces in 3D

- Blanz and Vetter (SIGGRAPH ’99)
  - <mpeg video>
  - The video can be found at:

    http://graphics.informatik.uni-freiburg.de/Sigg99.html

Tim Marks, Cognitive Science Department