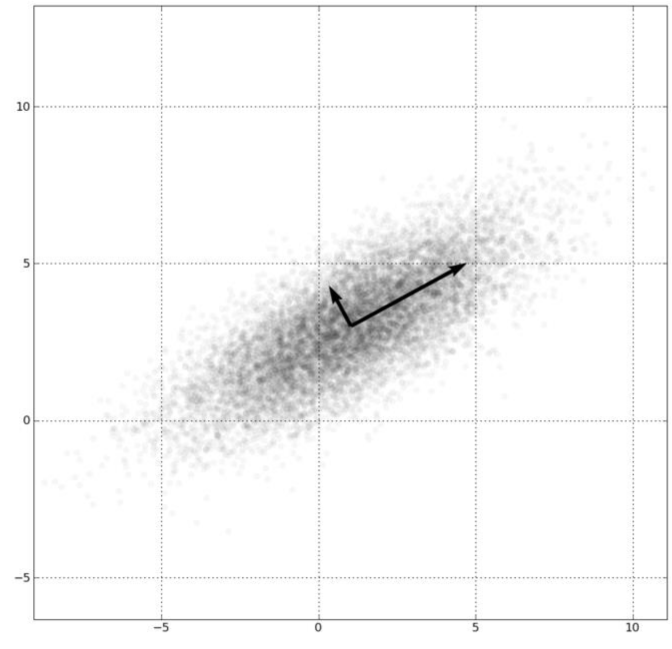
Unsupervised learning

- General introduction to unsupervised learning

PCA

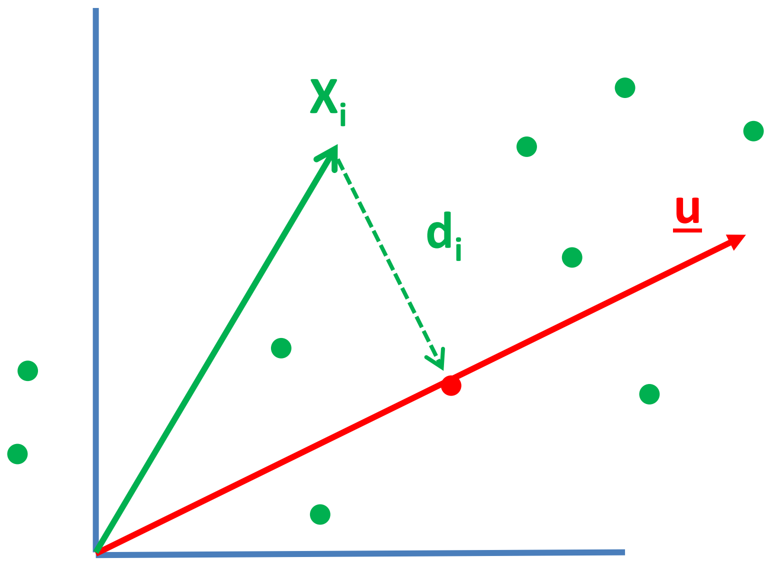
Special directions



These are special directions we will try to find.

Best direction $\underline{u}$:

$|\underline{u}|^2 = 1$



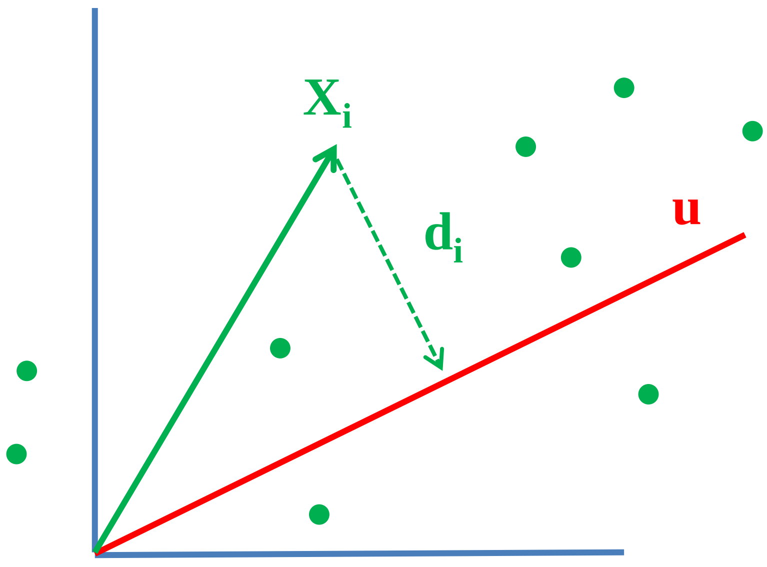
1. Minimize: Σd_i^2

$\underline{x}_i^T \underline{u}$ is the projection length

2. Maximize: $\Sigma (\underline{x}_i^T \underline{u})^2$

$\underline{u}$ is the direction that maximizes the variance

Finding the best projection:



Find u that maximize: $\Sigma\, (\underline{x}_i^T \underline{u})^2$

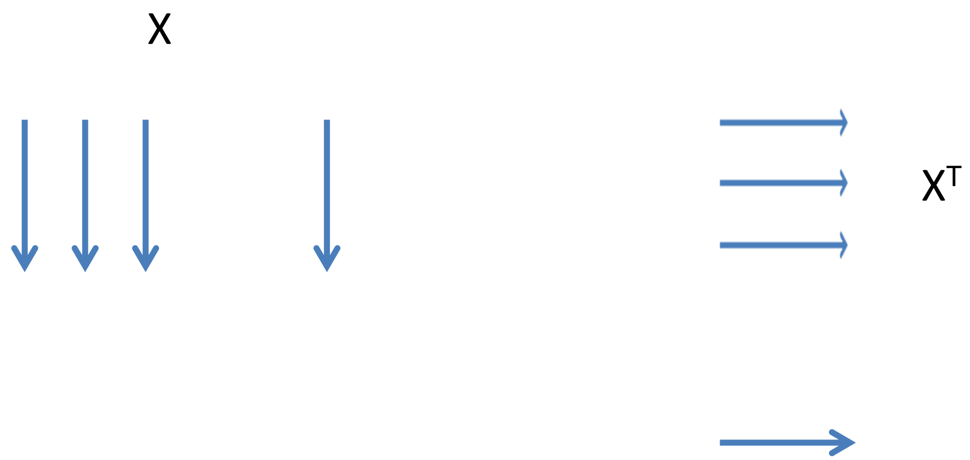
$(\underline{x}_i^T \underline{u})^2 \;=\; (\underline{u}^T \underline{x})\, (\underline{x}^T\, \underline{u})$

$\max \Sigma\, (\underline{u}^T \underline{x}_i)\, (\underline{x}_i^T\, \underline{u}) \quad = \quad \max \quad \underline{u}^T\, [V]\, u$

where: $[V] \;=\; \Sigma(\underline{x}_i\, \underline{x}_i^T)$

The data matrix:

[V] =



$$[V] = \Sigma(\underline{x}_i \underline{x}_i^T) = XX^T$$

Best direction $\underline{u}$

- Will minimize the distances from it
- Will maximize the variance along it

Max($\underline{u}$): $\underline{u}^T [V] \underline{u}$ subject to: $|u| = 1$

With Lagrange multipliers:

Maximize $\underline{u}^T [V] \underline{u} - \lambda(\underline{u}^T \underline{u} - 1)$

Derivative with respect to the vector $\underline{u}$:

$[V]\underline{u} - \lambda\underline{u} = 0$

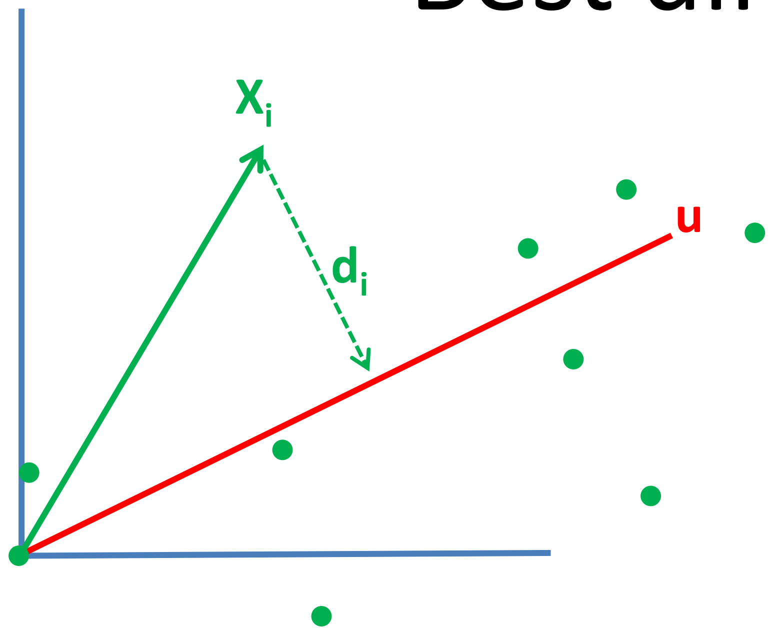
$[V]\underline{u} = \lambda\underline{u}$

$d/d\underline{x}\, (\underline{x}^T U \underline{x}) = 2U\underline{x}$

$d/d\underline{x}\, (\underline{x}^T \underline{x}) = 2\underline{x}$

The best direction will be the first eigenvector of [V]

Best direction u:



The best direction will be the first eigenvector of [V]; $\underline{u}_1$ with variance λ_1

The next direction will be the second eigenvector of [V]; $\underline{u}_2$ with variance λ_2

The Principle Components will be the eigenvectors of the data matrix

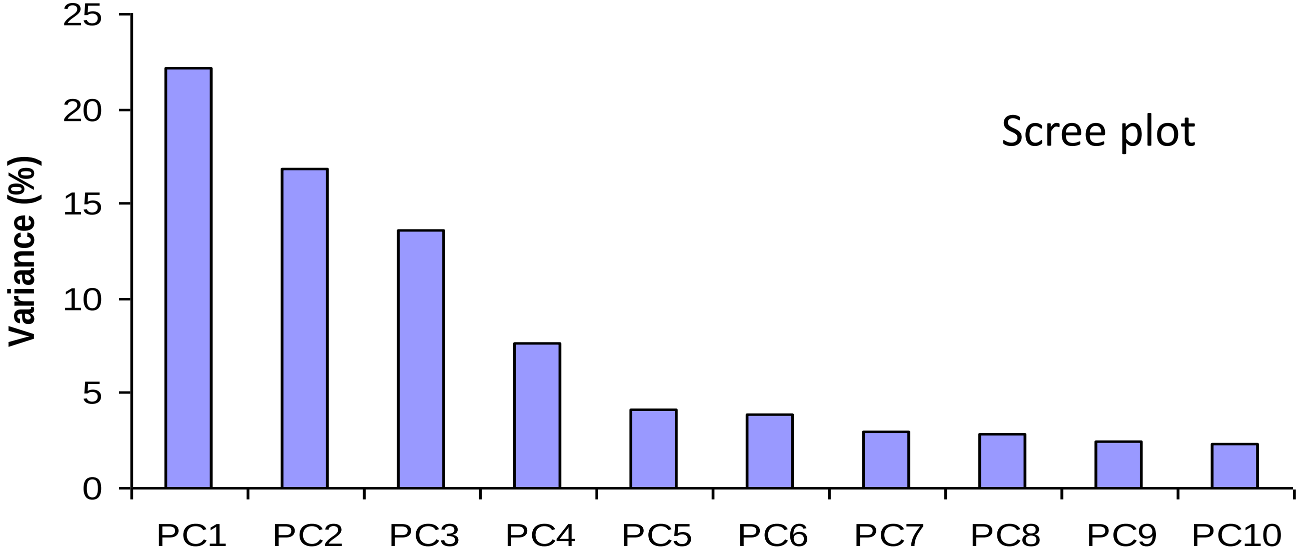
PCs, Variance and Least-Squares

- The first PC retains the greatest amount of variation in the sample
- The k^{th} PC retains the k^{th} greatest fraction of the variation in the sample
- The k^{th} largest eigenvalue of the correlation matrix C is the variance in the sample along the k^{th} PC

- The least-squares view: PCs are a series of linear least squares fits to a sample, each orthogonal to all previous ones

Dimensionality Reduction

Can *ignore* the components of lesser significance.

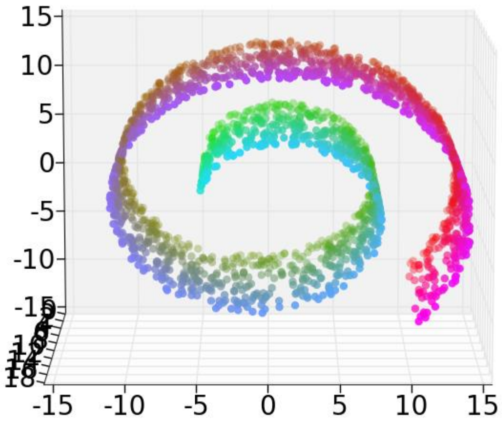


You do lose some information, but if the eigenvalues are small, you don't lose much

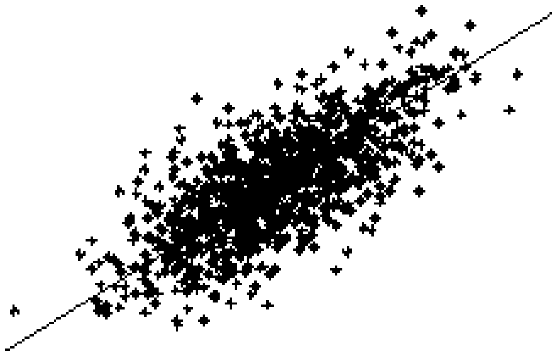
- n dimensions in original data
- calculate n eigenvectors and eigenvalues
- choose only the first k eigenvectors, based on their eigenvalues
- final data set has only k dimensions

PC dimensionality reduction

In the linear case only



PCA and correlations



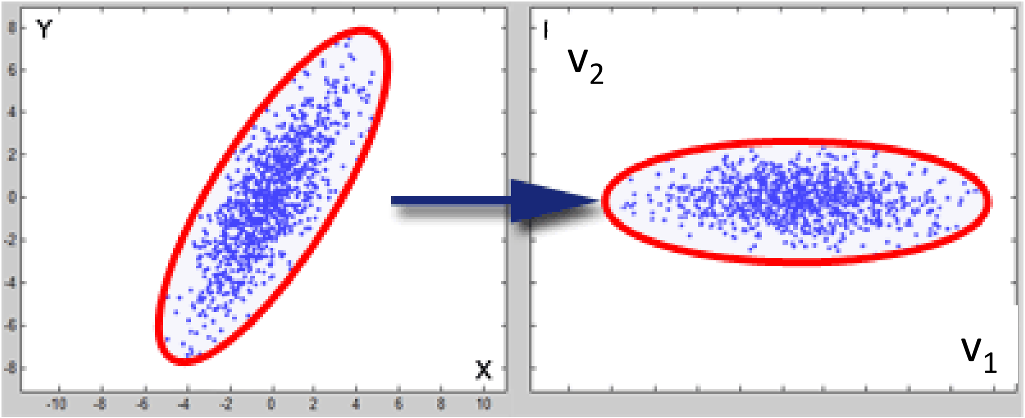
- We can think of our data points as k points from a distribution p($\underline{x}$)

- We have k samples (x_1 y_1) (x_2 y_2)… …(x_k y_k)

PCA and correlations

- We have k samples $(x_1\ y_1)\ (x_2\ y_2)\ldots\ \ldots(x_k\ y_k)$
- The correlation between(x,y) is: $E\ [\ (x-x_0)\ (y - y_0)\ /\ \sigma_x\ \sigma_y\ ]$

- For centered variables, x,y are uncorrelated if $E(xy) = 0$



Correlation depends on the coordinates:

(x,y) are correlated, (v_1 v_2) are not

In the PC coordinates, the variables are uncorrelated

- The projection of a point $x_{\mathbf{i}}$ on $\mathbf{v_1}$ is: $x_{\mathbf{i}}^{T}\mathbf{v_1}$ (or $\mathbf{v_1^T x_i}$).
- The projection of a point $x_{\mathbf{i}}$ on $\mathbf{v_2}$ is: $x_{\mathbf{i}}^{T}\mathbf{v_2}$
- For the correlation, we take the sum: $\Sigma_i (\mathbf{v_1^T x_i}) \ (x_{\mathbf{i}}^{T}\mathbf{v_2})$
- $= \quad \Sigma_i \mathbf{v_1^T x}_i \ \mathbf{x_i^T v_2} = \mathbf{v_1^T C v_2}$
- Where $C = X^T X$. (the data matrix)
- Since the v_i are eigenvectors of C, $\mathbf{C v_2} = \lambda_2 \mathbf{v_2}$
-
- $\mathbf{v_1^T C v_2} = \lambda_2 \ \mathbf{v_1^T v_2} = 0$
- The variables are uncorrelated.
- This is a result of using as coordinates the eigenvectors of the correlation matrix $C = X^T X$.

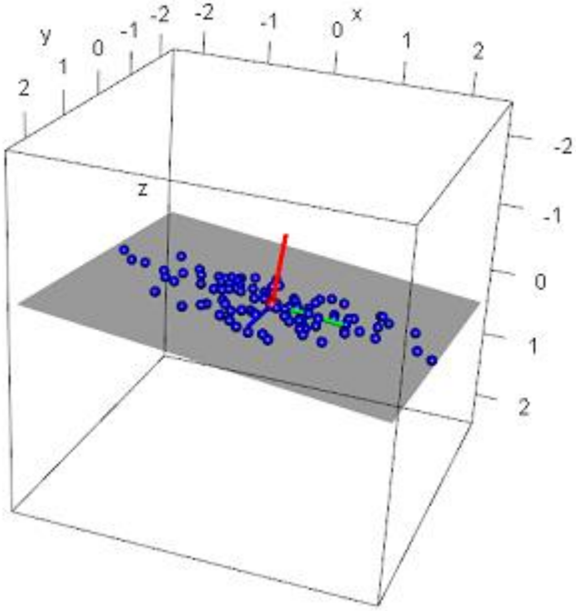
In the PC coordinates the variables are uncorrelated

- The correlation depends on the coordinate system. We can start with variables (x,y) which are correlated, transform them to (x', y') that will be un-correlated

- **If we use the coordinates defined by the eigenvectors of XX^T the variables (or the vectors x_i of n projections on the i'th axis) will be uncorrelated.**

Properties of the PCA

- The subspace spanned by the first k PC retains the maximal variance
- This subspace minimized the distance of the points from the subspace
- The transformed variables, which are linear combinations of the original ones, are uncorrelated.



Best plane, minimizing perpendicular distance over all planes

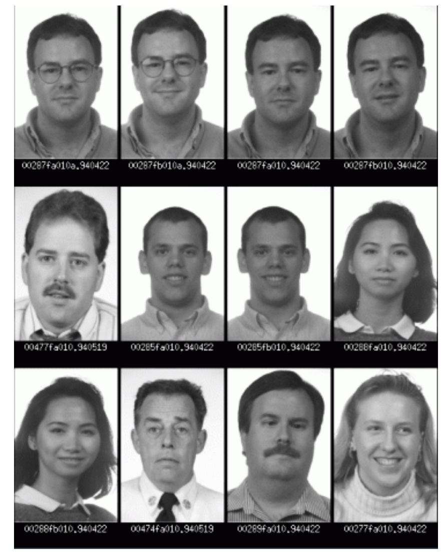
Eigenfaces: PC of face images

- Turk, M., Pentland, A.: *Eigenfaces for recognition*. J. Cognitive Neuroscience **3** (1991) 71–86.

Image Representation

- Training set of m images of size *N*N* are represented by vectors of size N^2

$$x_1, x_2, x_3, \ldots, x_M$$



Example

$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & -1 & 2 \\ 4 & 5 & 1 \end{bmatrix}_{3\times 3} \longrightarrow \begin{bmatrix} 1 \\ 2 \\ 3 \\ 3 \\ -1 \\ 2 \\ 4 \\ 5 \\ 1 \end{bmatrix}_{9\times 1}$$

Need to be well aligned

Average Image and Difference Images

- The average training set is defined by

$$\mu = (1/m) \sum^{m}_{i=1} x_i$$

- Each face differs from the average by vector

$$r_i = x_i - \mu$$

Covariance Matrix

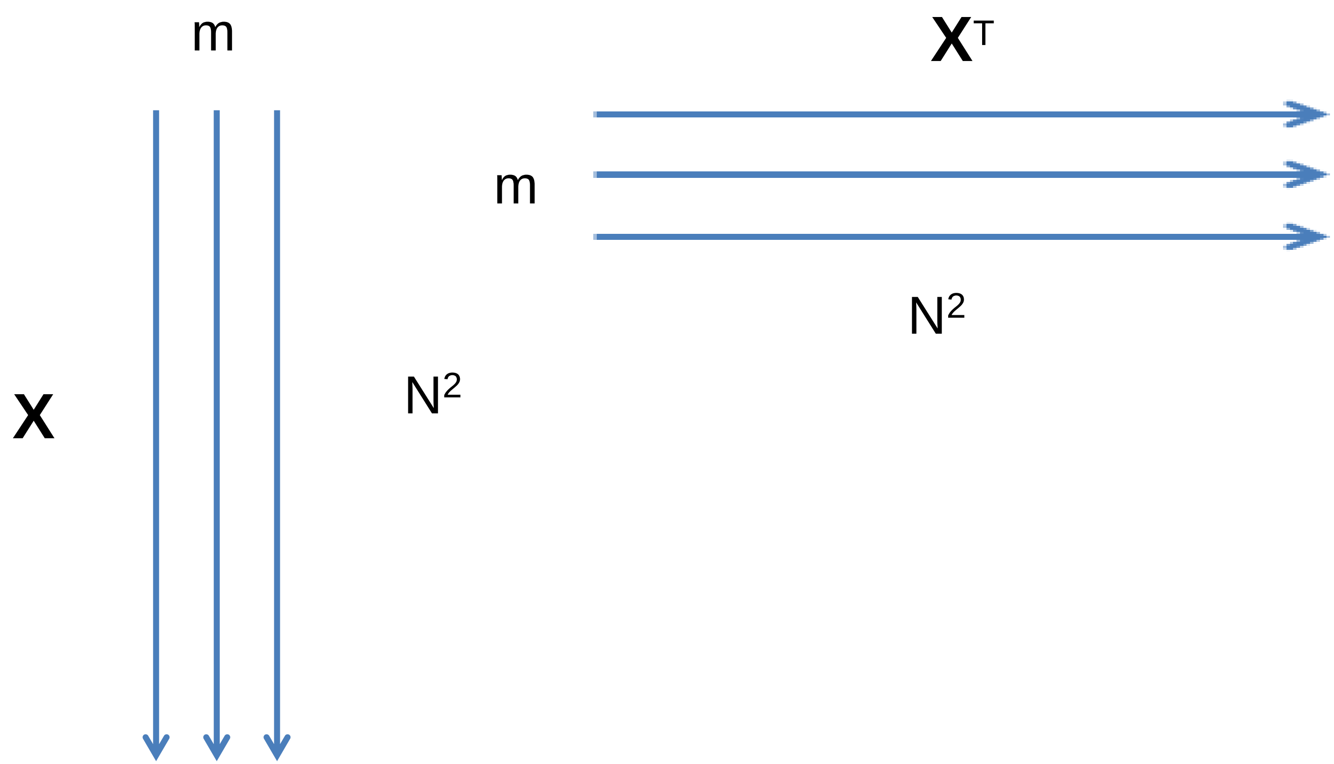
- The covariance matrix is constructed as

  $C = AA^T$ where $A=[r_1,\dots,r_m]$

  Size of this matrix is N^2 x N^2

- Finding eigenvectors of N^2 x N^2 matrix is intractable. Hence, use the matrix A^TA of size m x m and find eigenvectors of this small matrix.

Face data matrix:



XX^T is $N^2 * N^2$

X^TX is $m * m$

Eigenvectors of Covariance Matrix

- Consider the eigenvectors v_i of $A^T A$ such that

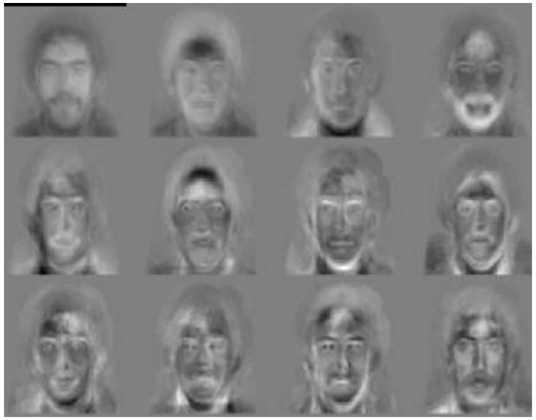
$$A^T A v_i = \mu_i v_i$$

- Pre-multiplying both sides by A, we have

$$AA^T (A v_i) = \mu_i (A v_i)$$

- $A v_i$ is an eigenvector of our original AA^T
- Find the eigenvectors v_i of the small $A^T A$
- Get the 'eigen-faces' by $A v_i$

Face Space



- u_i resemble facial images which look ghostly, hence called Eigenfaces

Projection into Face Space

- A face image can be projected into this face space by

$$p_k = U^T(x_k - \mu)$$

Rows of U^T are the eigenfaces

p_k are the m coefficients of face x_k

This is the representation of a face using eigen-faces

This representation can then be used for recognition using different recognition algorithms

Recognition in ‘face space’

- Turk and Pentland used 16 faces, and 7 pc
- In this case the face representation p:
- $p_k = U^T(x_k - \mu)$ is 7-long vector

- Face classification:
- Several images per face-class.
- For a new test image I: obtain the representation p_I
- Turk-Pentland used simple nearest neighbor
- Find NN in each class, take the nearest,
- s.t. distance $< \varepsilon$, otherwise result is ‘unknown’

- Other algorithms are possible, e.g. SVM

Face detection by 'face space'

- Turk-Pentland used 'faceness' measure:

- Within a window, compare the original image with its reconstruction from face-space

- Find the distance ϵ between the original image x and its reconstructed image from the eigenface space, x_f,
  $\epsilon^2 = \| x - x_f \|^2$, where $x_f = U\underline{p} + \mu$ (reconstructed face)

- If $\varepsilon < \theta$ for a threshold θ
- A face is detected in the window

- Not 'state-of-the-art and not fast enough
- Eigenfaces in the brain?

Next: PCA by Neurons