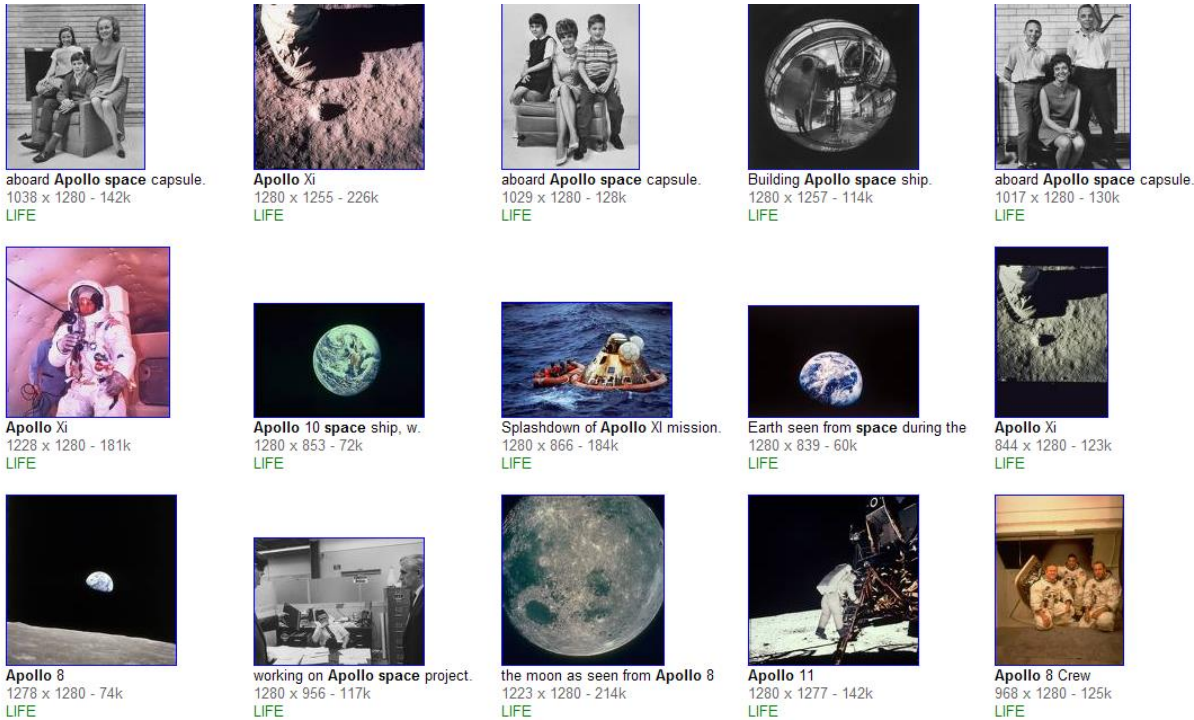
Machine Learning for Signal Processing

Data driven representations:
1. Eigenfaces and Eigenrepresentations

Class 5. 15 Sep 2015

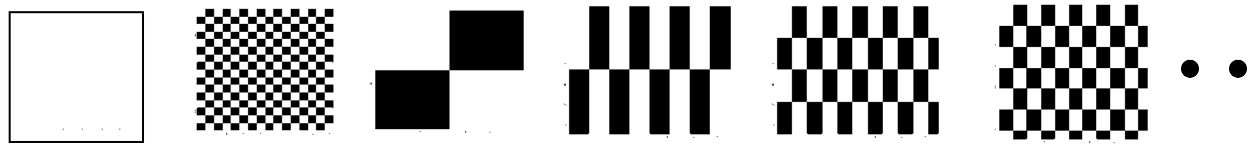
Instructor: Bhiksha Raj

Recall: Representing images



- The most common element in the image: background
  - Or rather large regions of relatively featureless shading
  - Uniform sequences of numbers

Adding more bases



B_1 B_2 B_3 B_4 B_5 B_6 ..

- Checkerboards with different variations

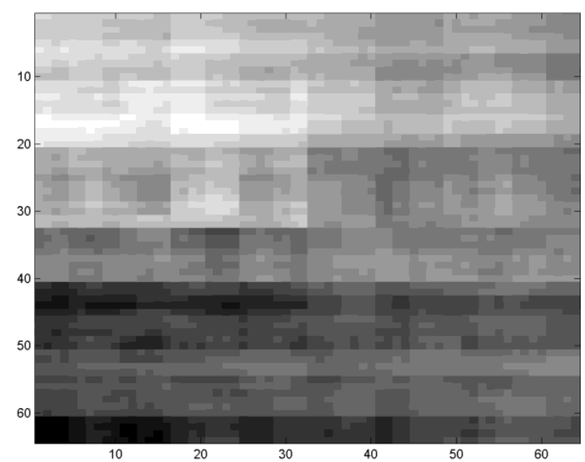
$$\text{Image} \approx w_1 B_1 + w_2 B_2 + w_3 B_3 + \ldots$$

$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ . \\ . \end{bmatrix} \qquad B = [B_1 \;\; B_2 \;\; B_3]$$

$$BW \approx \text{Image}$$

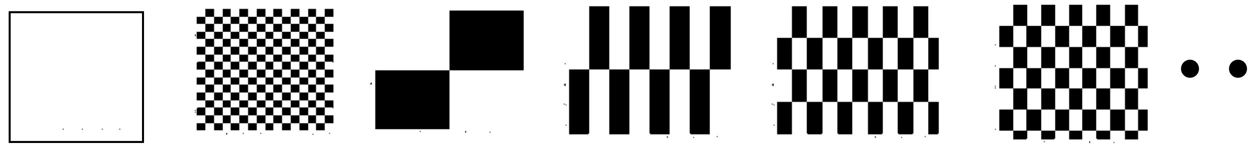
$$W = pinv(B)\,\text{Image}$$

$$PROJECTION = BW$$



Getting closer at 625 bases!

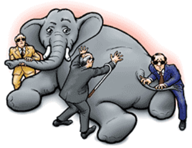
"Bases"



B_1 B_2 B_3 B_4 B_5 B_6

$$image \approx w_1 B_1 + w_2 B_2 + w_3 B_3 + \ldots$$

- "Bases" are the "standard" units such that all instances can be expressed a weighted combinations of these units
- Ideal requirements: Bases must be orthogonal
- Checkerboards are one choice of bases
  - Orthogonal
  - But not "smooth"
- Other choices of bases: Complex exponentials, Wavelets, etc..



Data specific bases?

- **Issue**: **The bases we have considered so far are *data agnostic***
  - Checkerboards, Complex exponentials, Wavelets..
  - We use the same bases regardless of the data we analyze
    - Image of face vs. Image of a forest
    - Segment of speech vs. Seismic rumble
- How about data specific bases
  - Bases that consider the underlying data
    - E.g. is there something better than checkerboards to describe faces
    - Something better than complex exponentials to describe music?

The Energy Compaction Property

- **Define "better"?**
- The description

$$X = w_1 B_1 + w_2 B_2 + w_3 B_3 + \ldots + w_N B_N$$

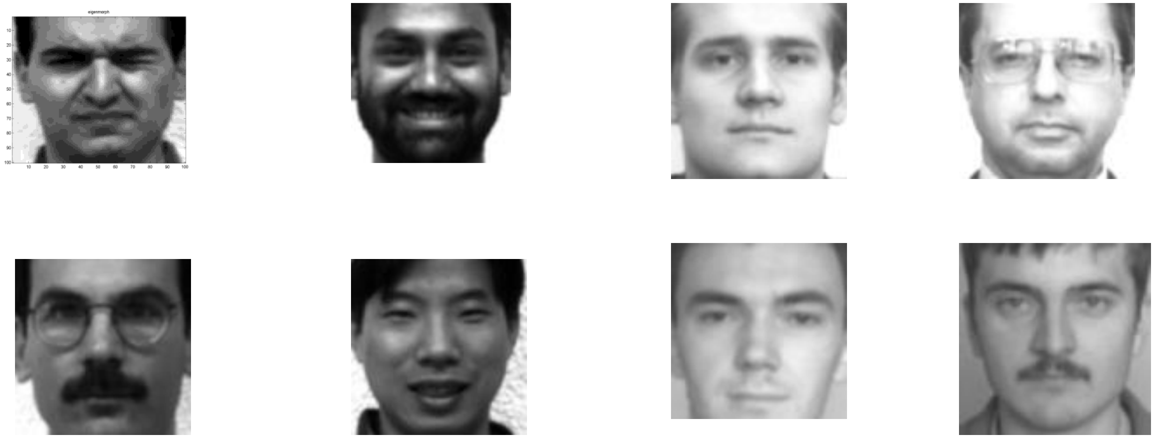
- The ideal:

$$\hat{X}_i \approx w_1 B_1 + w_2 B_2 + \ldots + w_i B_i \qquad Error_i = \left\| X - \hat{X}_i \right\|^2$$

$$Error_i < Error_{i-1}$$

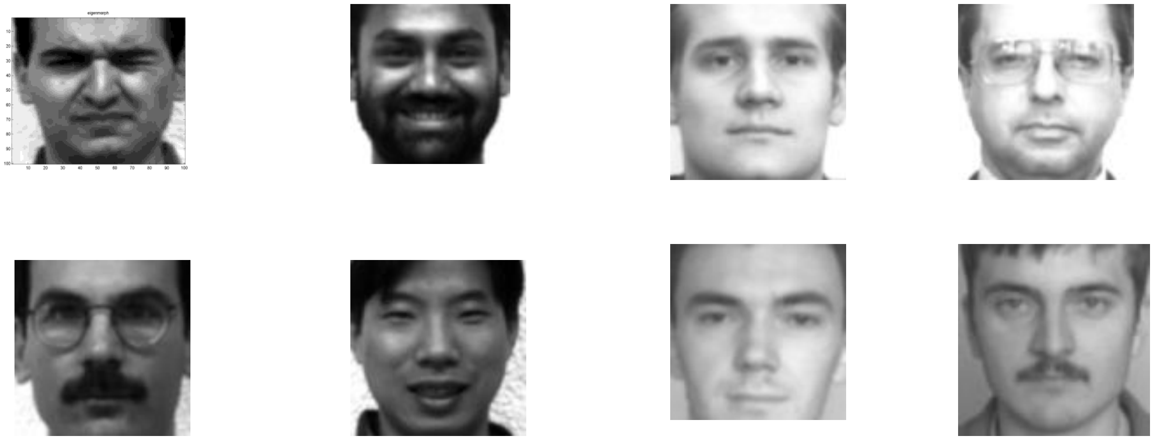
  – If the description is terminated at any point, we should still get most of the information about the data
    - Error should be small

Data-specific description of faces

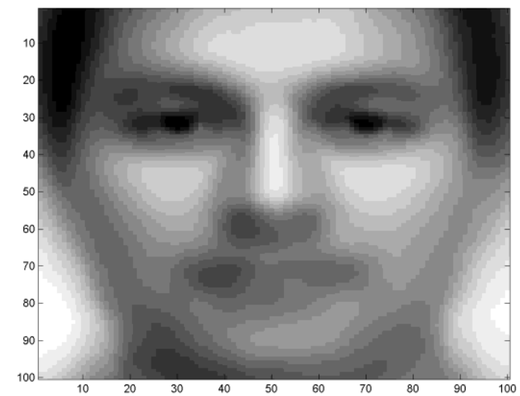


- A collection of images
  - All normalized to 100x100 pixels
- What is common among all of them?
  - Do we have a common descriptor?

A typical face

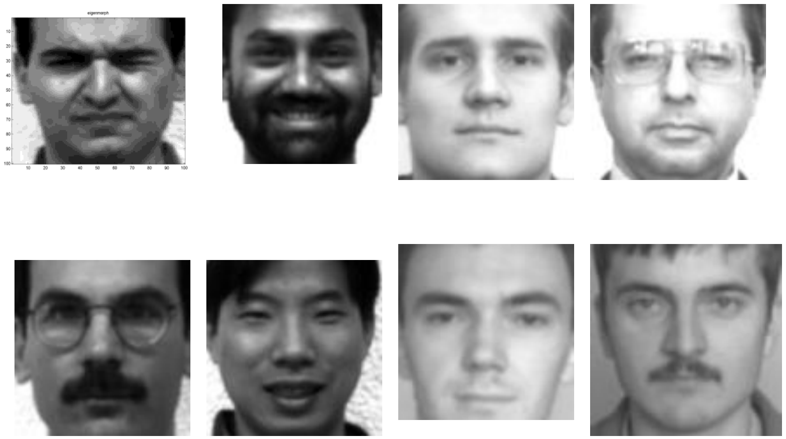


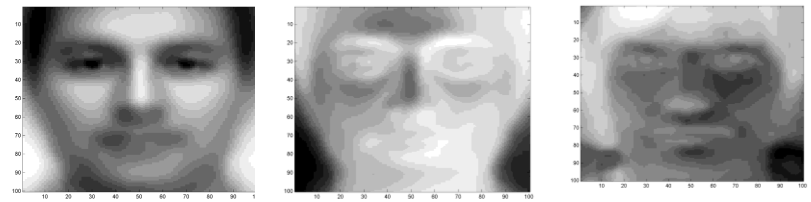
The typical face



- **Assumption: There is a "typical" face that captures most of what is common to all faces**
  - Every face can be represented by a scaled version of a typical face
  - We will denote this face as V
- Approximate **every** face f as $f = w_f V$
- Estimate V to minimize the squared error
  - How? What is V?

A collection of least squares typical faces





..

- Assumption: There are a set of *K* “typical” faces that captures most of all faces
- Approximate **every** face f as $f = w_{f,1} V_1 + w_{f,2} V_2 + w_{f,3} V_3 + .. + w_{f,k} V_k$
  - V_2 is used to “correct” errors resulting from using only V_1. So on average

$$\left\| f - (w_{f,1}V_{f,1} + w_{f,2}V_{f,2}) \right\|^2 < \left\| f - w_{f,1}V_{f,1} \right\|^2$$

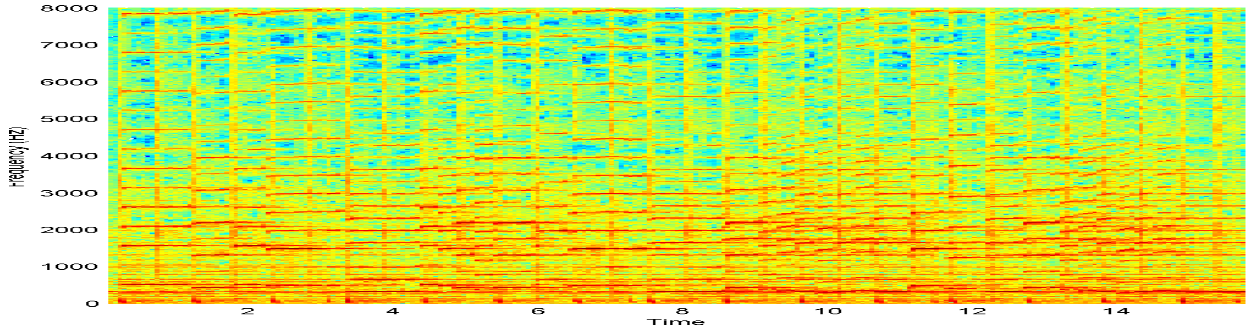
  - V_3 corrects errors remaining after correction with V_2

$$\left\| f - (w_{f,1}V_{f,1} + w_{f,2}V_{f,2} + w_{f,3}V_{f,3}) \right\|^2 < \left\| f - (w_{f,1}V_{f,1} + w_{f,2}V_{f,2}) \right\|^2$$

  - And so on..
  - **$V = [V_1\ V_2\ V_3]$**
- Estimate V to minimize the squared error
  - ***How? What is V?***

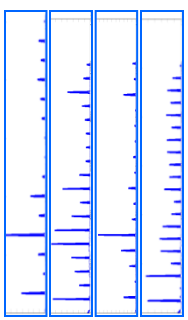
A recollection

M =



S=Pinv(N) M

N =



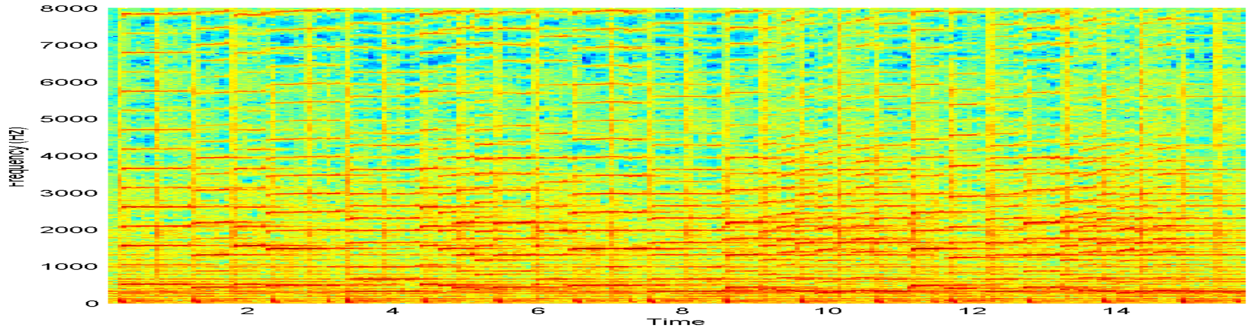
$$U = NS \approx M$$

$$S = pinv(N)M$$

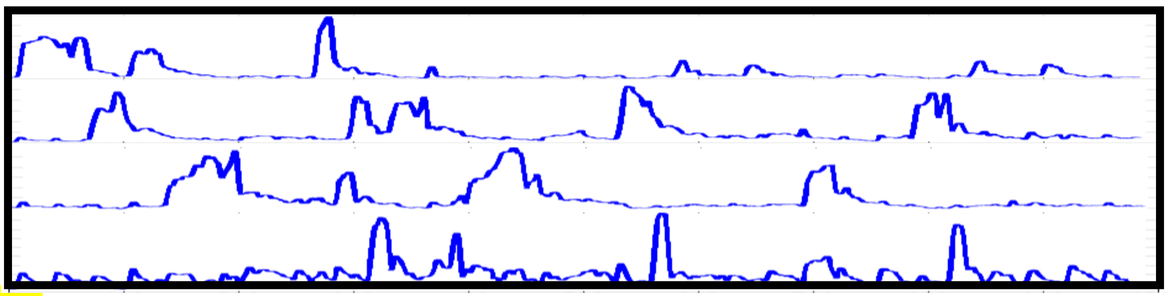
- Finding the best explanation of music M in terms of notes N
- Also finds the *score* S of M in terms of N

How about the other way?

M =



S =



N = M Pinv(S)

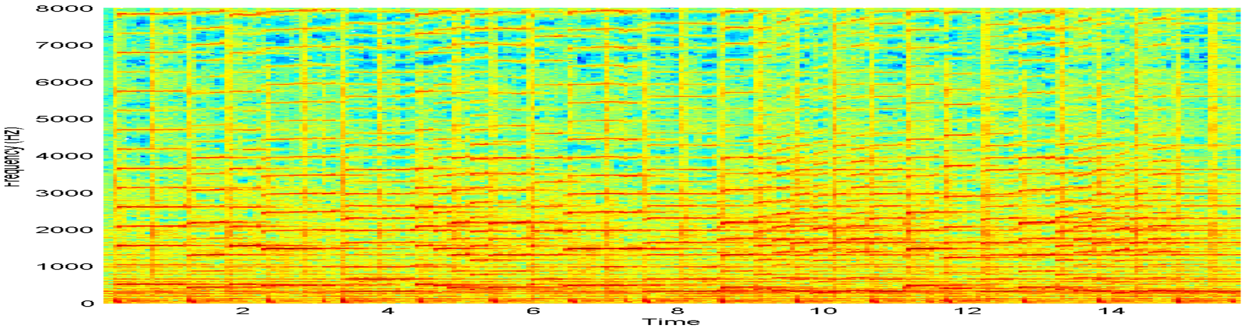
N = ?

U = ?

$$U = NS \approx M$$
$$N = M \ pinv(S)$$

- Finding the *notes* N given music M and score S
- Also finds best explanation of M in terms of S

Finding Everything

M = 

S = ?

N = ?

U = ?

$$U = NS \approx M$$

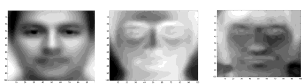
- Find the four notes and their score that generate the closest approximation to M

The same problem

F =

W

Typical faces

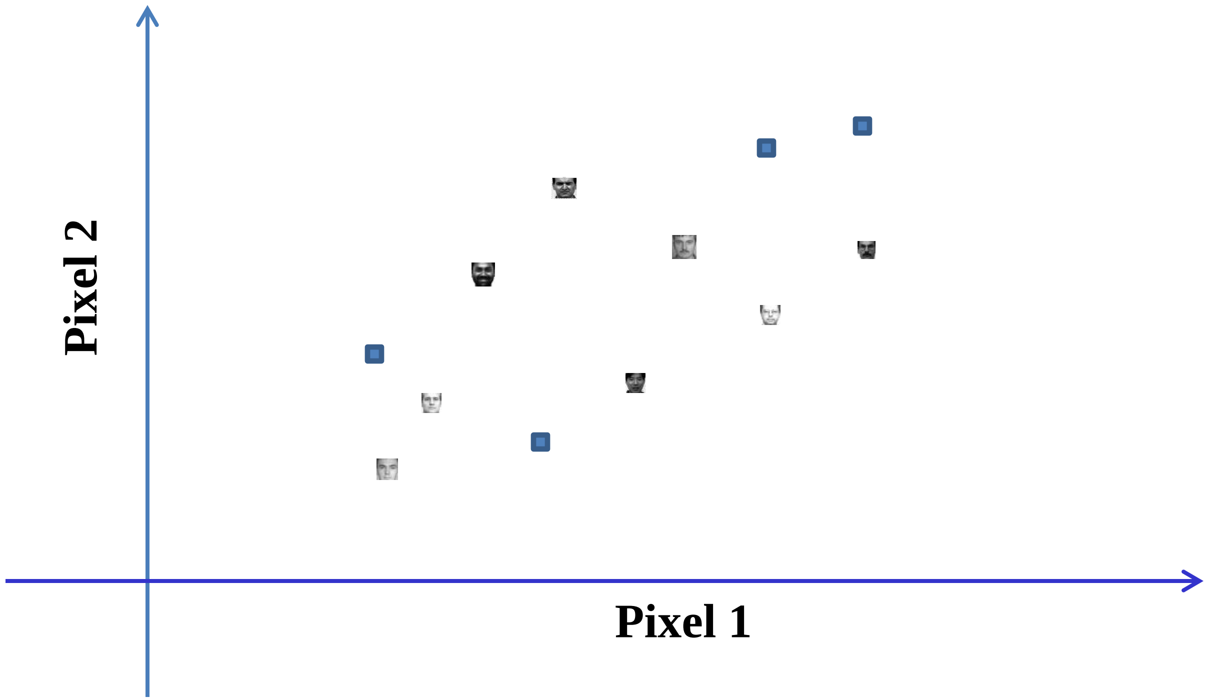


V

U = Approximation

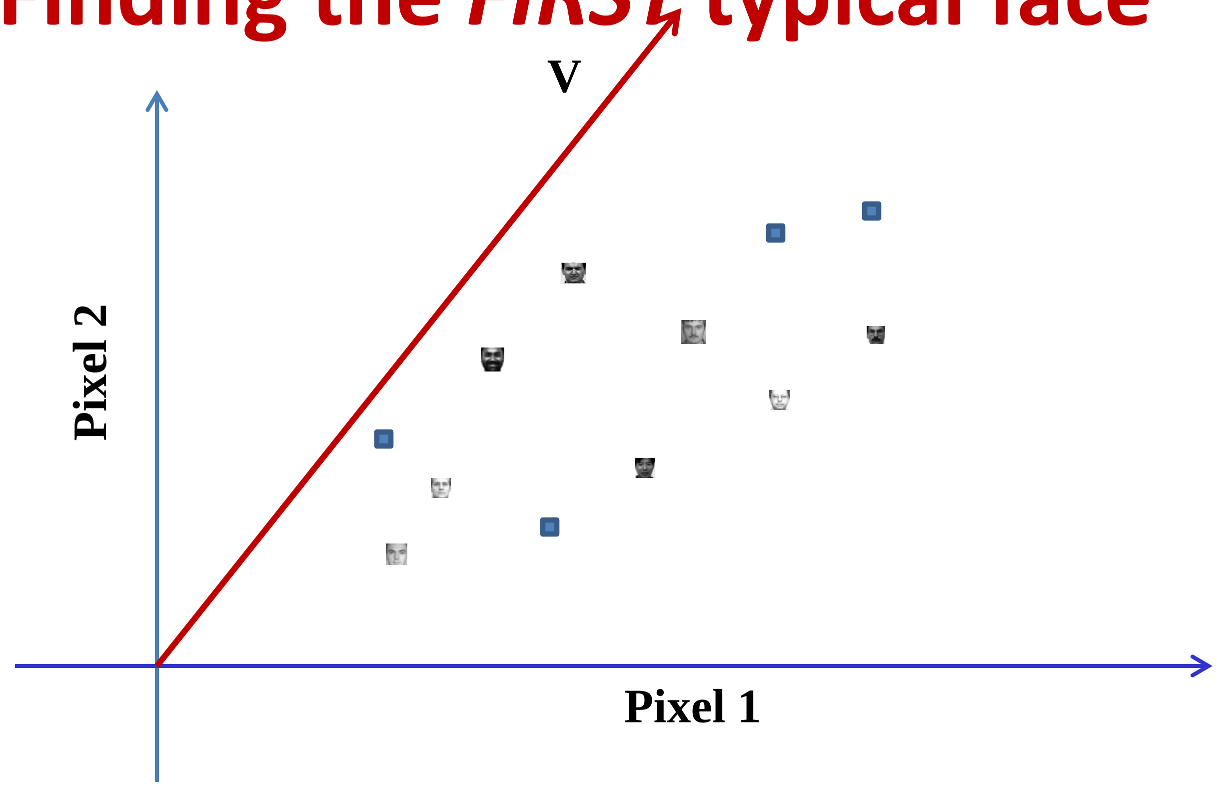
- Here V, W and U are ALL unknown and must be determined
  - Such that the squared error between U and F is minimum
- For each face
  - $f = w_{f,1} V_1 + w_{f,2} V_2 + w_{f,3} V_3 + .. + w_{f,K} V_K$
- For the collection of faces: $F \approx V\, W$
  - V is $D \times K$ and W is $K \times N$
    - D is the no. of pixels, N, is the no. of faces in the set

Abstracting the problem: Finding the *FIRST* typical face



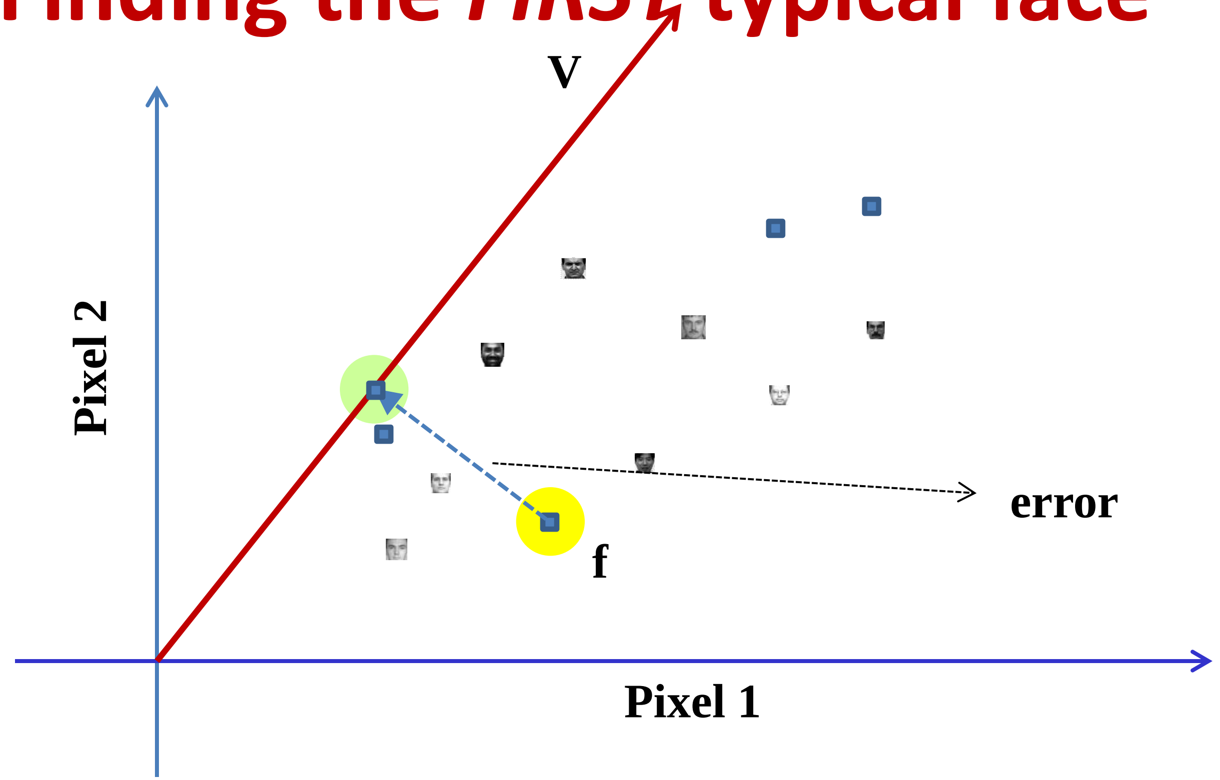
- Each "point" represents a face in "pixel space"

Abstracting the problem:
Finding the *FIRST* typical face



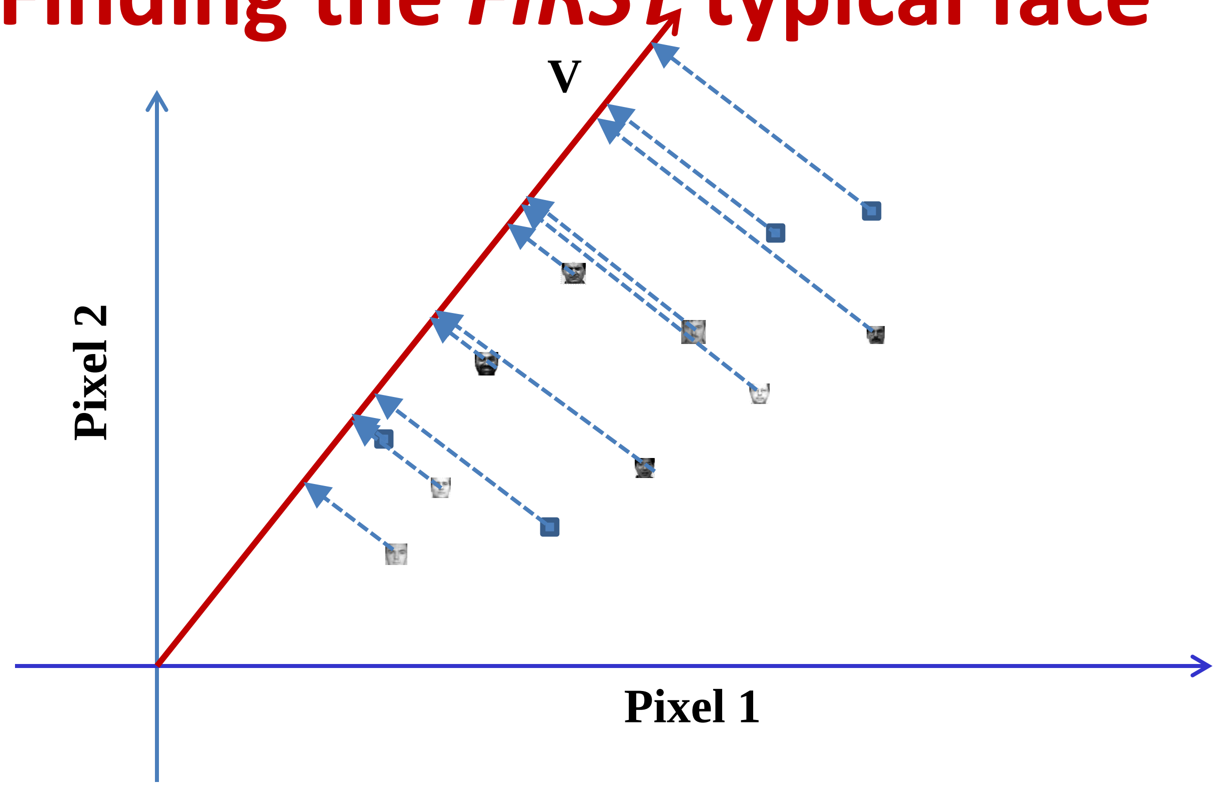
- Each "point" represents a face in "pixel space"
- Any "typical face" V is a vector in this space

Abstracting the problem: Finding the *FIRST* typical face



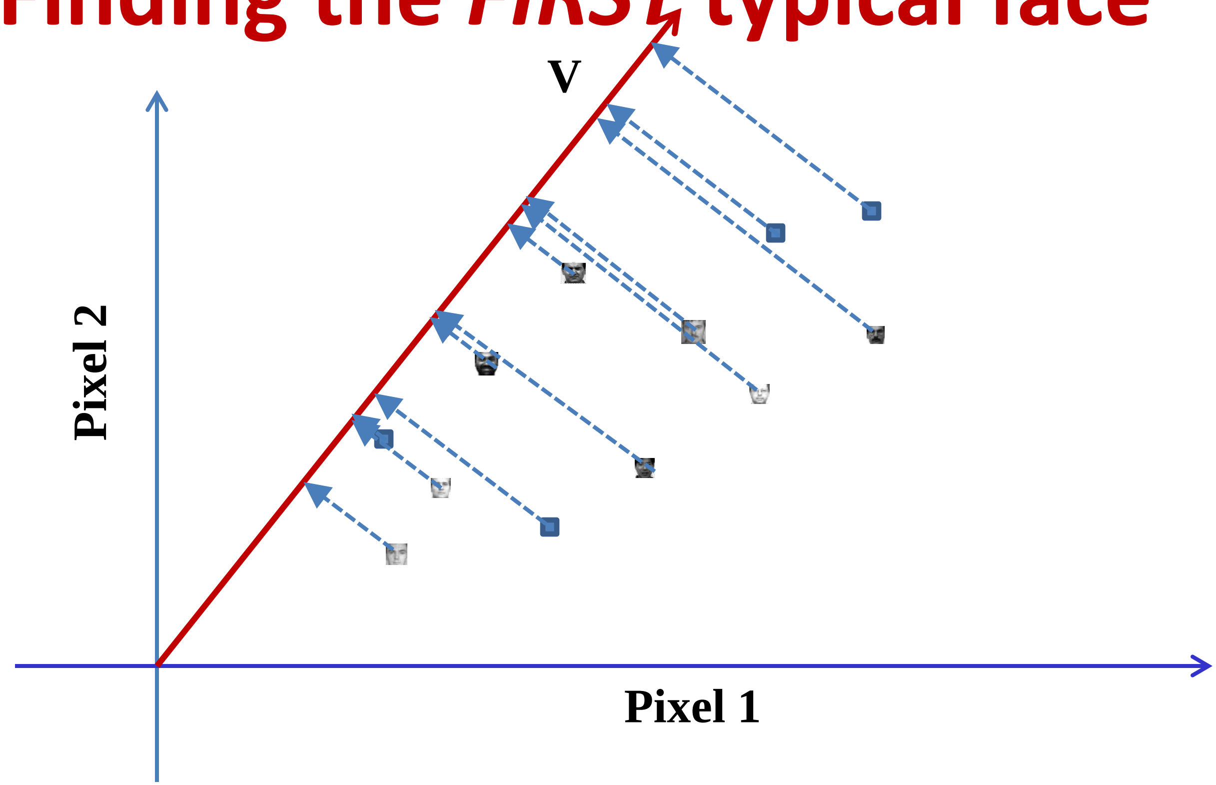
- Each "point" represents a face in "pixel space"
- The "typical face" V is a vector in this space
- The ***approximation*** $w_{f,}\ V$ for any face f is the *projection* of f onto V
- The distance between f and its projection $w_f V$ is the *projection error* for f

Abstracting the problem:
Finding the *FIRST* typical face



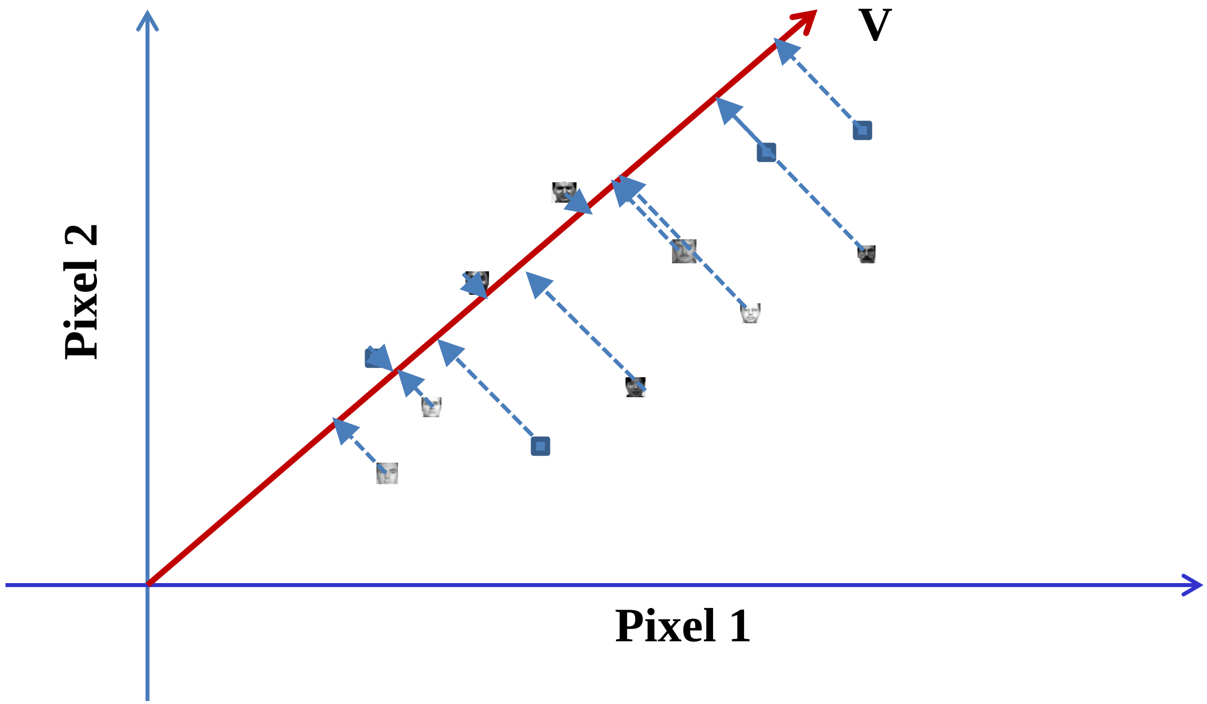
- *Every* face in our data will suffer error when approximated by its projection on V
- The total squared length of all error lines is the *total squared projection error*

Abstracting the problem: Finding the *FIRST* typical face



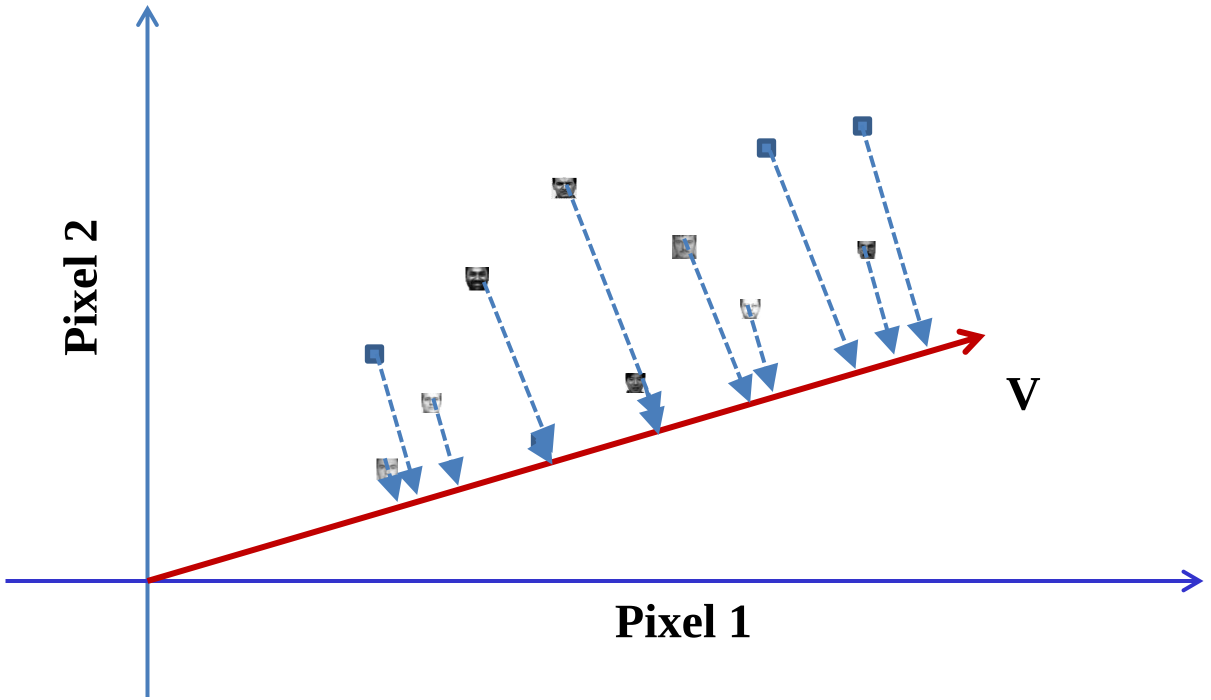
- The problem of finding the first typical face V_1:
  Find the V for which the total projection error is minimum!

Abstracting the problem:
Finding the *FIRST* typical face



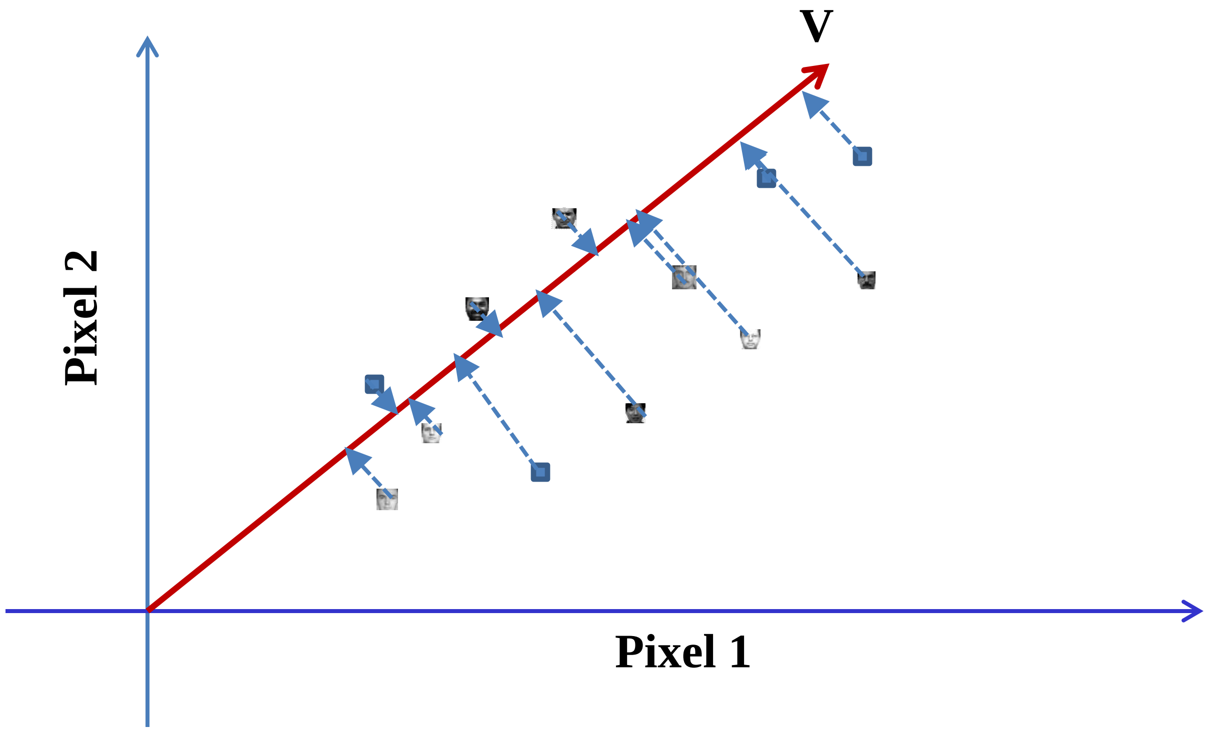
- The problem of finding the first typical face V_1:
  Find the V for which the total projection error is minimum!

Abstracting the problem:
Finding the *FIRST* typical face



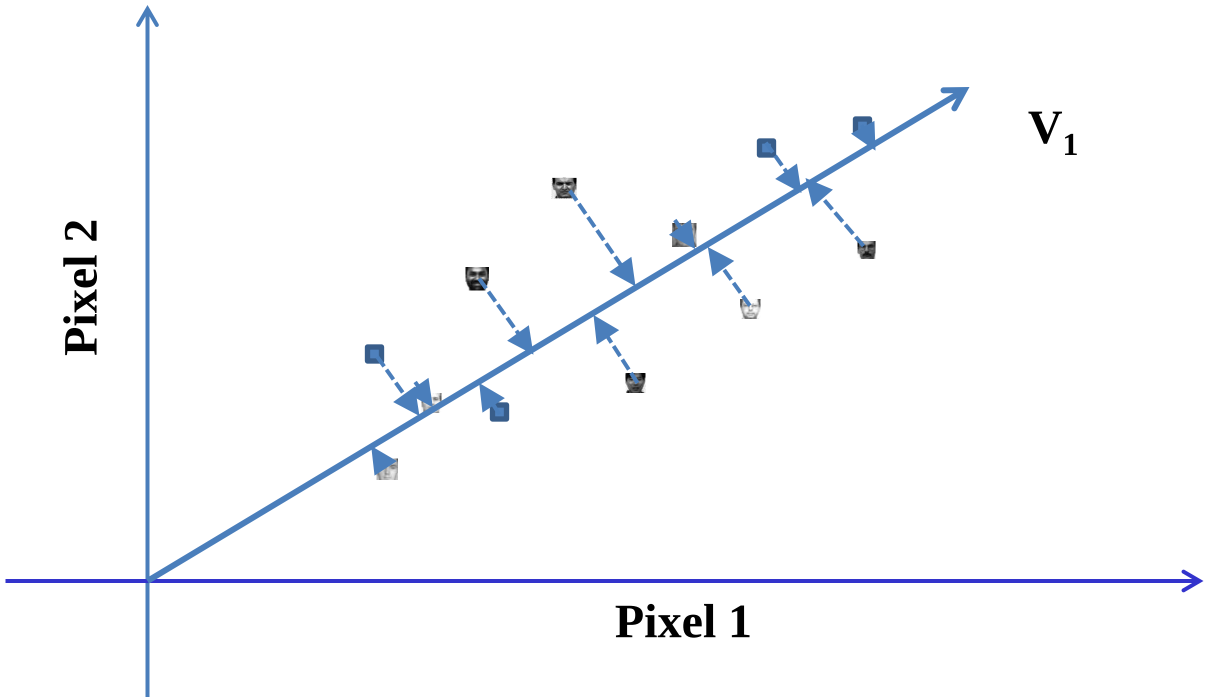
- The problem of finding the first typical face V_1:
  Find the V for which the total projection error is minimum!

Abstracting the problem:
Finding the *FIRST* typical face



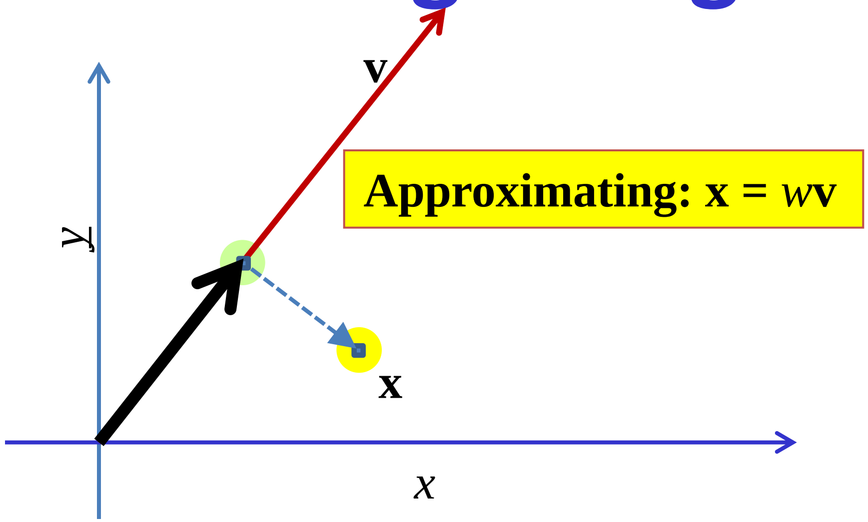
- The problem of finding the first typical face V_1:
  Find the V for which the total projection error is minimum!

Abstracting the problem: Finding the *FIRST* typical face



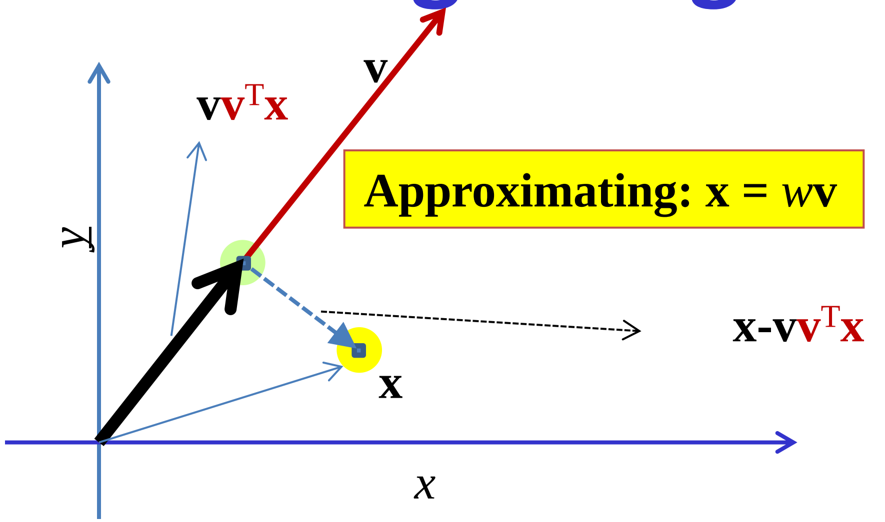
- The problem of finding the first typical face V_1:
  Find the V for which the total projection error is minimum!
- This "minimum squared error" V is our "best" first typical face
- **It is also the first *Eigen face***

Formalizing the Problem: Error from approximating a single vector



- Consider: approximating $\mathbf{x} = w\mathbf{v}$
  - E.g **x** is a face, and "**v**" is the "typical face"
- Finding an approximation $w\mathbf{v}$ which is closest to **x**
  - In a Euclidean sense
  - Basically projecting **x** onto **v**

Formalizing the Problem: Error from approximating a single vector



- Projection of a vector $\mathbf{x}$ on to a vector $\mathbf{v}$

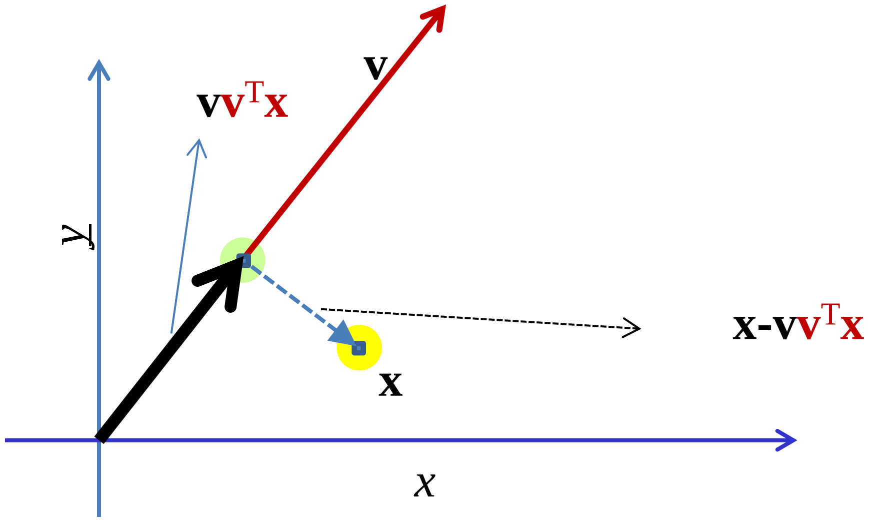
$$\hat{\mathbf{x}} = \mathbf{v}\frac{\mathbf{v}^T\mathbf{x}}{|\mathbf{v}|^2}$$

- Assuming $\mathbf{v}$ is of unit length: $\hat{\mathbf{x}} = \mathbf{w}^T\mathbf{x}$

$$error = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{w}^T\mathbf{x}$$

$$\text{squared error} = \left\|\mathbf{x} - \mathbf{w}^T\mathbf{x}\right\|^2$$

Error from approximating a single vector

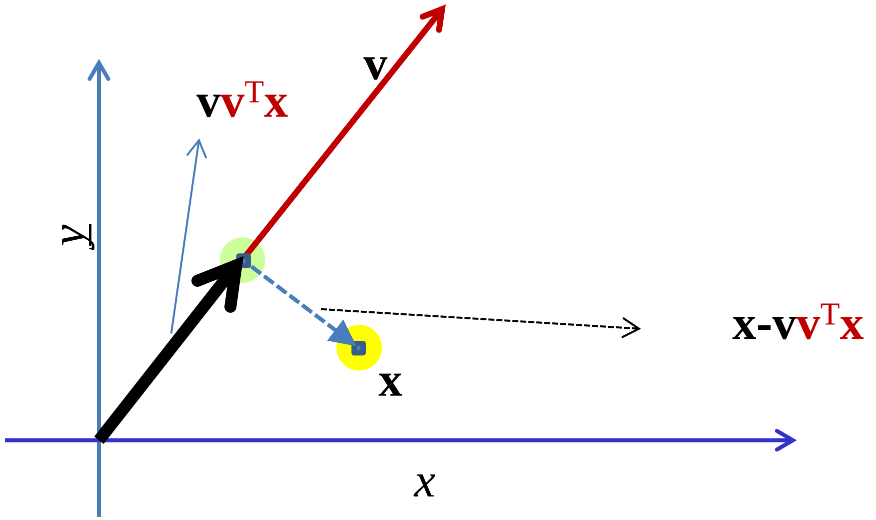


- Minimum squared approximation error from approximating **x** as it as $w\mathbf{v}$

$$e(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{w}^T \mathbf{x} \right\|^2$$

- Optimal value of w: $w = \mathbf{v}^T\mathbf{x}$

Error from approximating a single vector

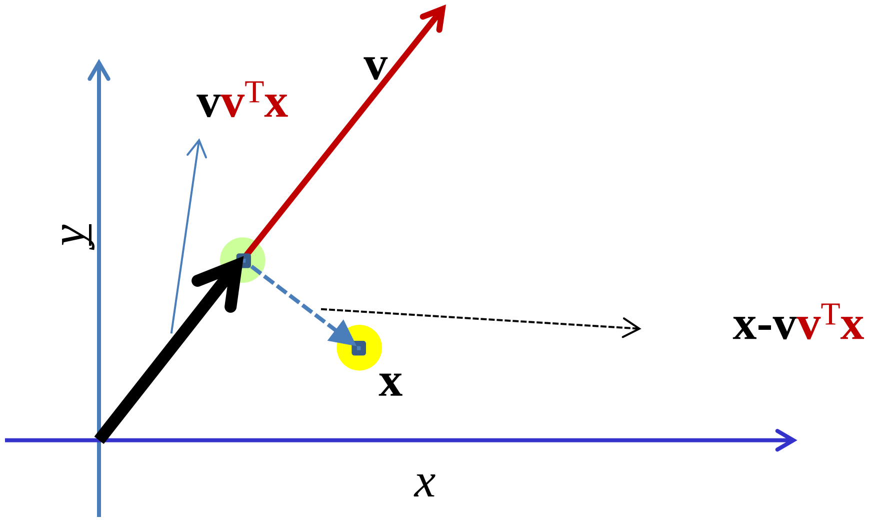


- Error from projecting a vector $\mathbf{x}$ on to a vector onto a unit vector $\mathbf{v}$ $e(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{w}^T\mathbf{x} \right\|^2$

$$e(\mathbf{x}) = (\mathbf{x} - \mathbf{w}^T\mathbf{x})^T(\mathbf{x} - \mathbf{w}^T\mathbf{x}) = (\mathbf{x}^T - \mathbf{x}^T\mathbf{w}^T)(\mathbf{x} - \mathbf{w}^T\mathbf{x})$$

$$= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{w}^T\mathbf{w}^T\mathbf{x}$$

Error from approximating a single vector



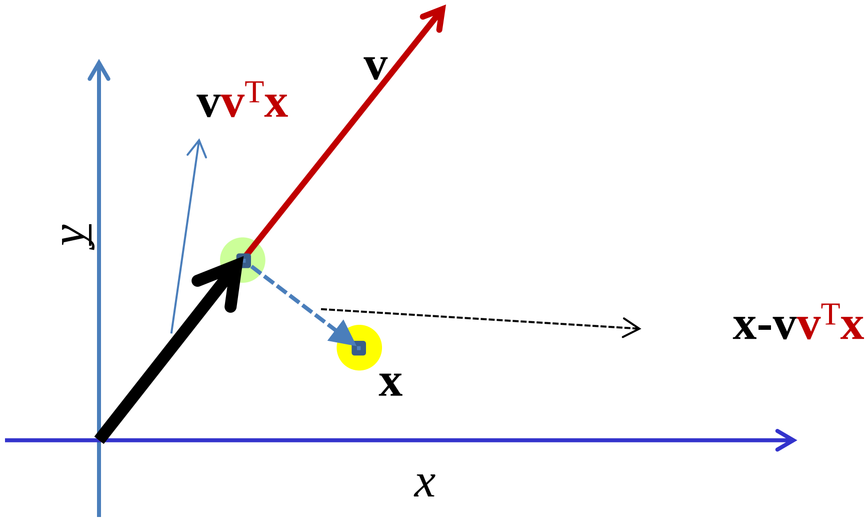
- Error from projecting a vector **x** on to a vector onto a unit vector **v**

$$e(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{w}^T\mathbf{x} \right\|^2$$

$$e(\mathbf{x}) = \left(\mathbf{x} - \mathbf{w}^T\mathbf{x}\right)^T\left(\mathbf{x} - \mathbf{w}^T\mathbf{x}\right) = \left(\mathbf{x}^T - \mathbf{x}^T\mathbf{w}^T\right)\left(\mathbf{x} - \mathbf{w}^T\mathbf{x}\right)$$

$$= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} + \mathbf{x}^T\underbrace{\mathbf{w}^T\mathbf{w}}_{=1}{}^T\mathbf{x}$$

Error from approximating a single vector



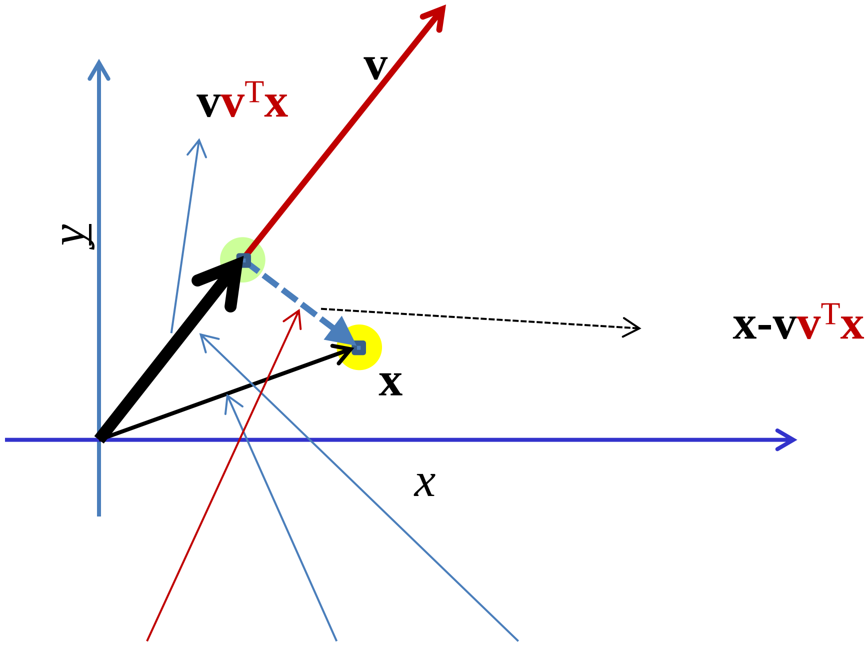
- Error from projecting a vector **x** on to a vector onto a unit vector **v** $e(\mathbf{x}) = \left\| \mathbf{x} - \mathbf{w}^T\mathbf{x} \right\|^2$

$$e(\mathbf{x}) = (\mathbf{x} - \mathbf{w}^T\mathbf{x})^T(\mathbf{x} - \mathbf{w}^T\mathbf{x}) = (\mathbf{x}^T - \mathbf{x}^T\mathbf{w}^T)(\mathbf{x} - \mathbf{w}^T\mathbf{x})$$

$$= \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x} + \mathbf{x}^T\mathbf{w}^T\mathbf{x}$$

$$e(\mathbf{x}) = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x}$$

Error from approximating a single vector

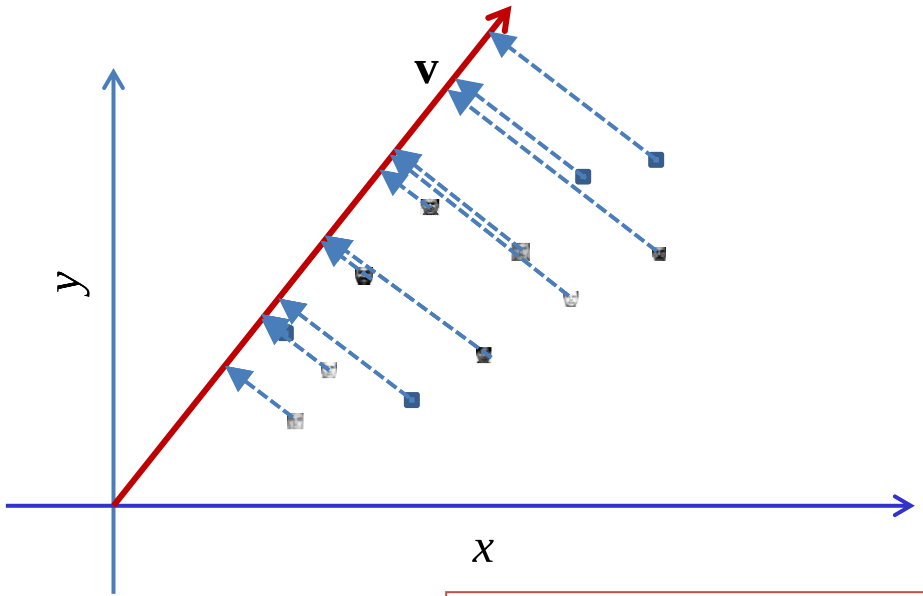


$$\mathbf{e}(\mathbf{x}) = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{v}.\mathbf{v}^T\mathbf{x}$$

Length of projection

This is the very familiar pythogoras' theorem!!

Error for *many* vectors

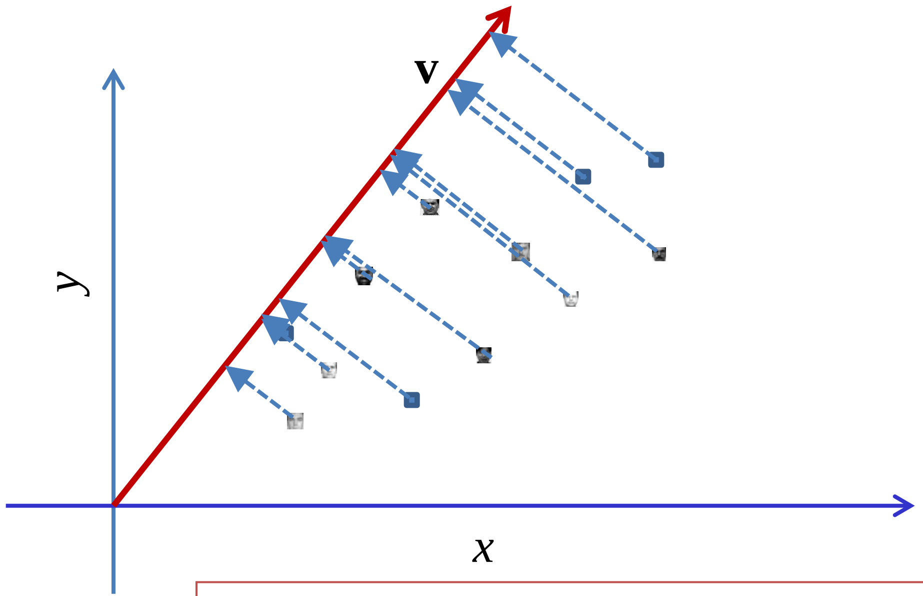


- Error for one vector: $e(\mathbf{x}) = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{w}^T\mathbf{x}$
- Error for many vectors

$$E = \sum_i e(\mathbf{x}_i) = \sum_i \left(\mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{w}^T\mathbf{x}_i\right) = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{w}^T\mathbf{x}_i$$

- **Goal: Estimate v to minimize this error!**

Error for *many* vectors



- Total error: $$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{w}^T \mathbf{x}_i$$

- Add constraint: $\mathbf{v}^T\mathbf{v} = 1$

- Constrained objective to minimize:

$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{w}^T \mathbf{x}_i + \lambda\left(\mathbf{v}^T \mathbf{v} - 1\right)$$

Two Matrix Identities

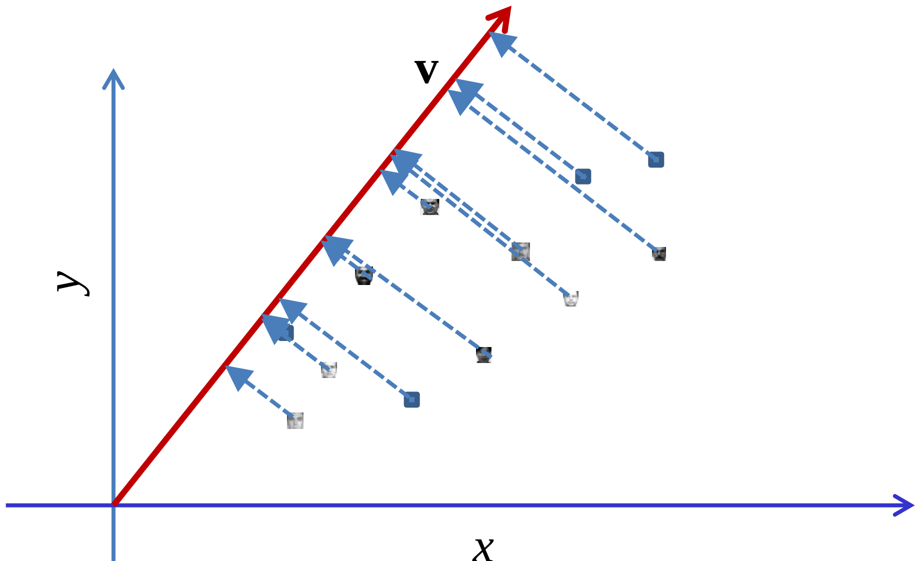
- Derivative w.r.t $\mathbf{v}$

$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{w}^T \mathbf{x}_i + \lambda\left(\mathbf{v}^T \mathbf{v} - 1\right)$$

$$\frac{d\mathbf{v}^T \mathbf{v}}{d\mathbf{v}} = 2\mathbf{v}$$

$$\frac{d\mathbf{x}^T \mathbf{w}^T \mathbf{x}}{d\mathbf{v}} = \frac{d\mathbf{v}^T \mathbf{x}\mathbf{x}^T \mathbf{v}}{d\mathbf{v}} = 2\mathbf{x}\mathbf{x}^T \mathbf{v}$$

Minimizing error



$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{w}^T \mathbf{x}_i + \lambda\left(\mathbf{v}^T\mathbf{v} - 1\right)$$

- Differentiating w.r.t $\mathbf{v}$ and equating to 0

$$-2\sum_i \mathbf{x}_i \mathbf{x}_i^T \mathbf{v} + 2\lambda\mathbf{v} = 0$$

$$\left(\sum_i \mathbf{x}_i \mathbf{x}_i^T\right)\mathbf{v} = \lambda\mathbf{v}$$

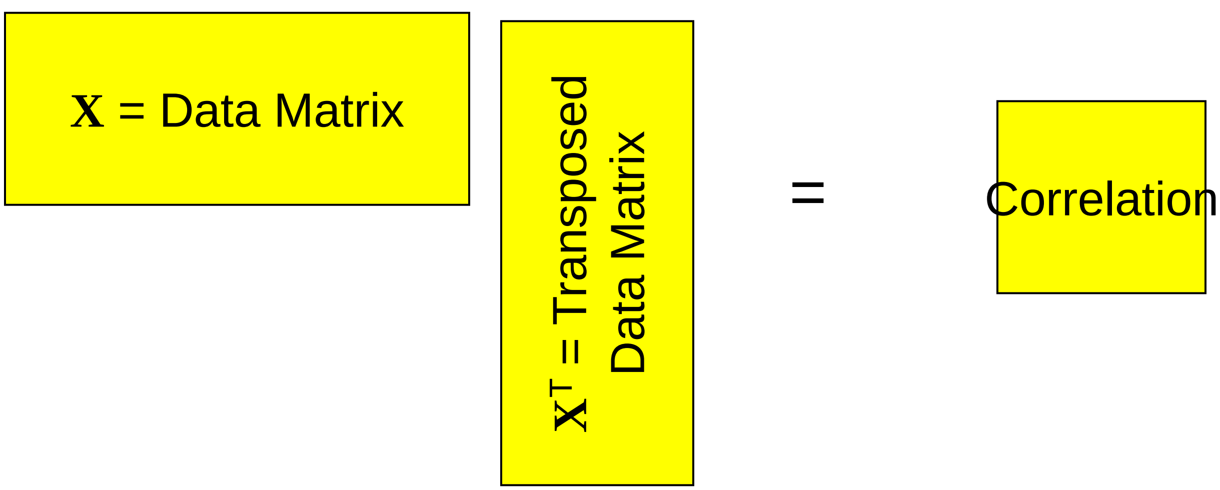
The correlation matrix

$$\left(\sum_i \mathbf{x}_i \mathbf{x}_i^T\right)\mathbf{v} = \lambda \mathbf{v}$$

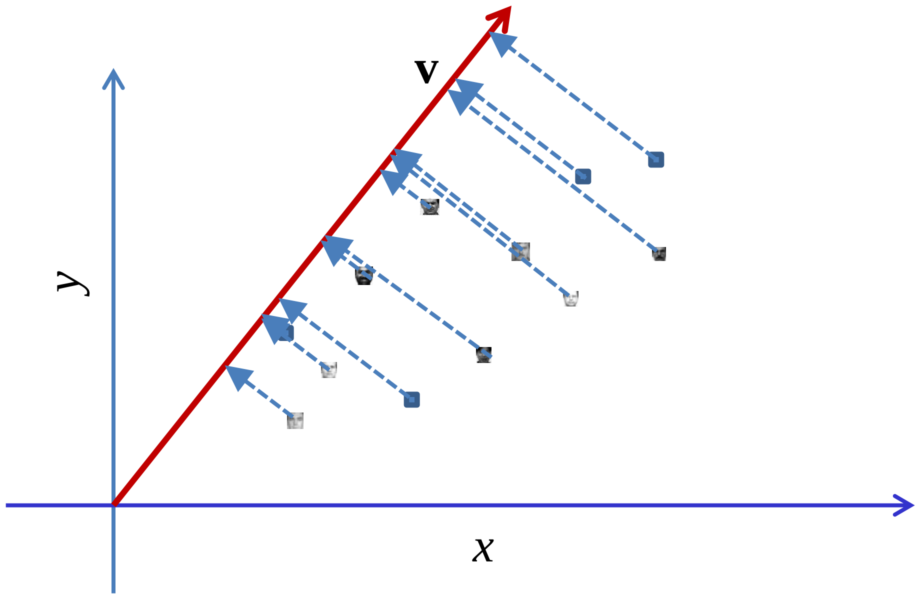
- The encircled term is the *correlation matrix*

$$\mathbf{X} = \left[\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \mathbf{x}_N\right]$$

$$\sum_i \mathbf{x}_i \mathbf{x}_i^T = \mathbf{X}\mathbf{X}^T = \mathbf{R}$$



The best "basis"



- The minimum-error basis is found by solving

$$\mathbf{Rv} = \lambda \mathbf{v}$$

- $\mathbf{v}$ is an Eigen vector of the correlation matrix $\mathbf{R}$
  - λ is the corresponding Eigen value

What about the total error?

$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{v}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{v}$$

- $\mathbf{x}^{\mathrm{T}}\mathbf{v} = \mathbf{v}^{\mathrm{T}}\mathbf{x}$ (inner product)

$$= \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \left(\sum_i \mathbf{x}_i \mathbf{x}_i^T \right) \mathbf{v}$$

$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \mathbf{R} \mathbf{v} = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \lambda \mathbf{v} = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \lambda \mathbf{v}^T \mathbf{v}$$

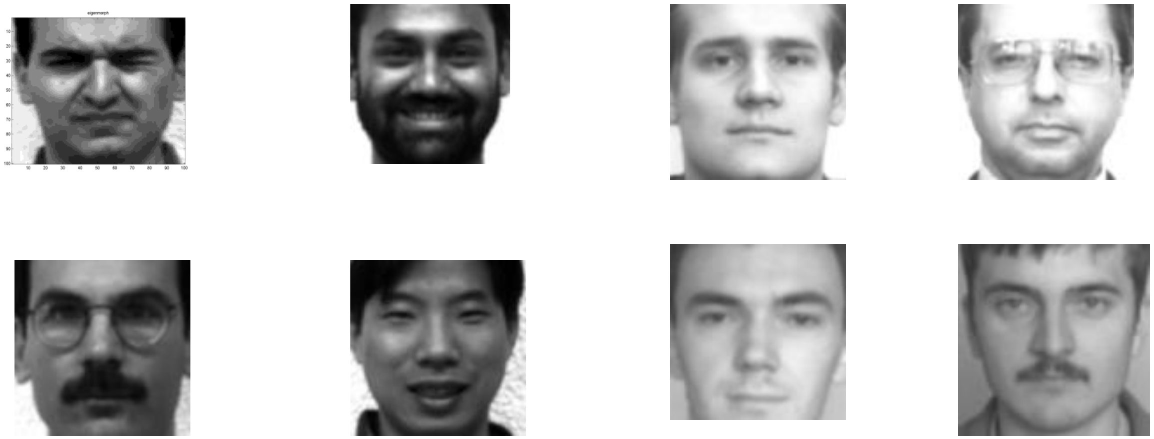
$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \lambda$$

Minimizing the error

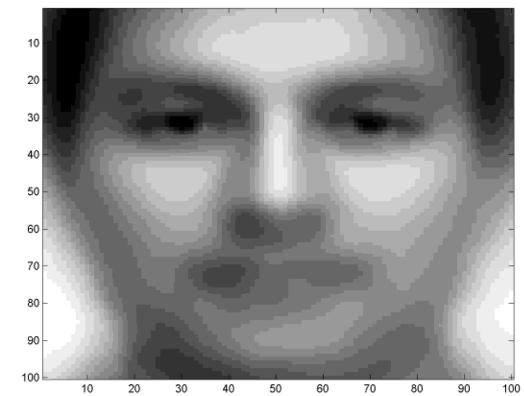
- The total error is $E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \lambda$

- We already know that the optimal basis is an Eigen vector
- The total error depends on the *negative* of the corresponding Eigen value
- To *minimize* error, we must *maximize* λ
- i.e. Select the Eigen vector with the largest Eigen value

The typical face



The typical face

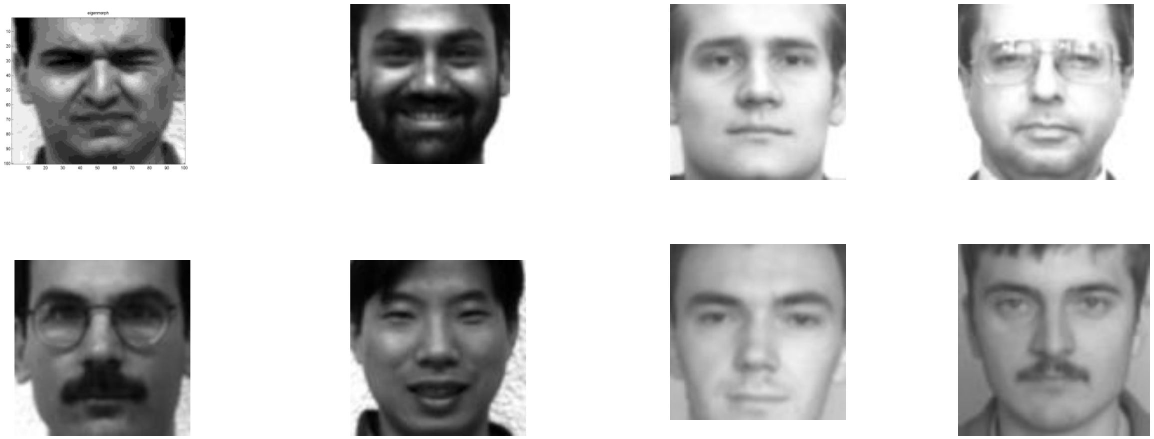


- Compute the correlation matrix for your data
  - Arrange them in matrix **X** and compute **R** = $\mathbf{XX}^T$
- Compute the *principal* Eigen vector of R
  - The Eigen vector with the largest Eigen value
- This is the typical face

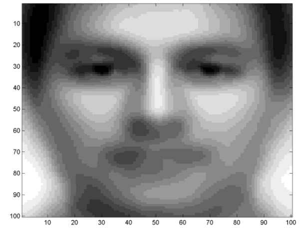
The *approximation* with the first typical face

- The first typical face models some of the characteristics of the faces
  - Simply by scaling its grey level
- But the approximation has error
- The *second* typical face must explain some of this error

The *second* typical face



The *first* typical face

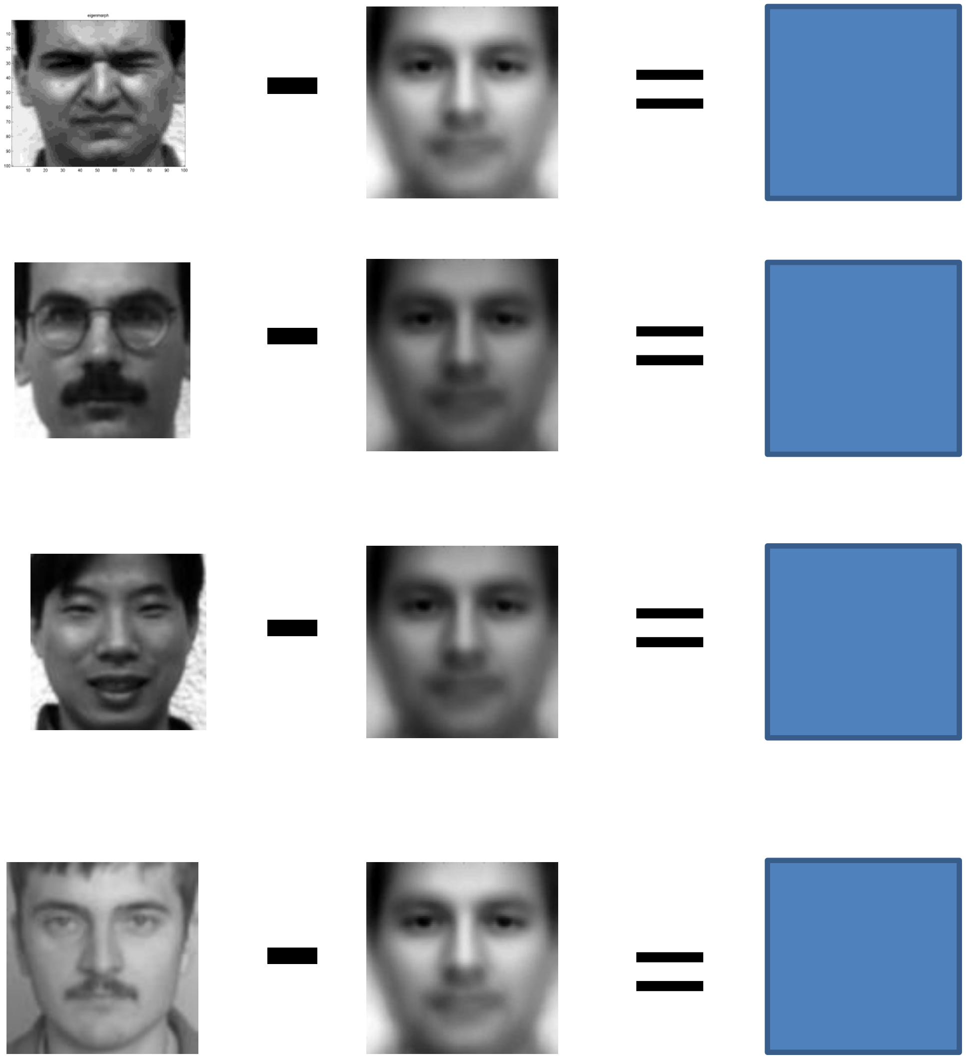


The *second* typical face?

?

- Approximation with only the first typical face has error
- The *second* face must explain this error
- How do we find this this face?

Solution: Iterate

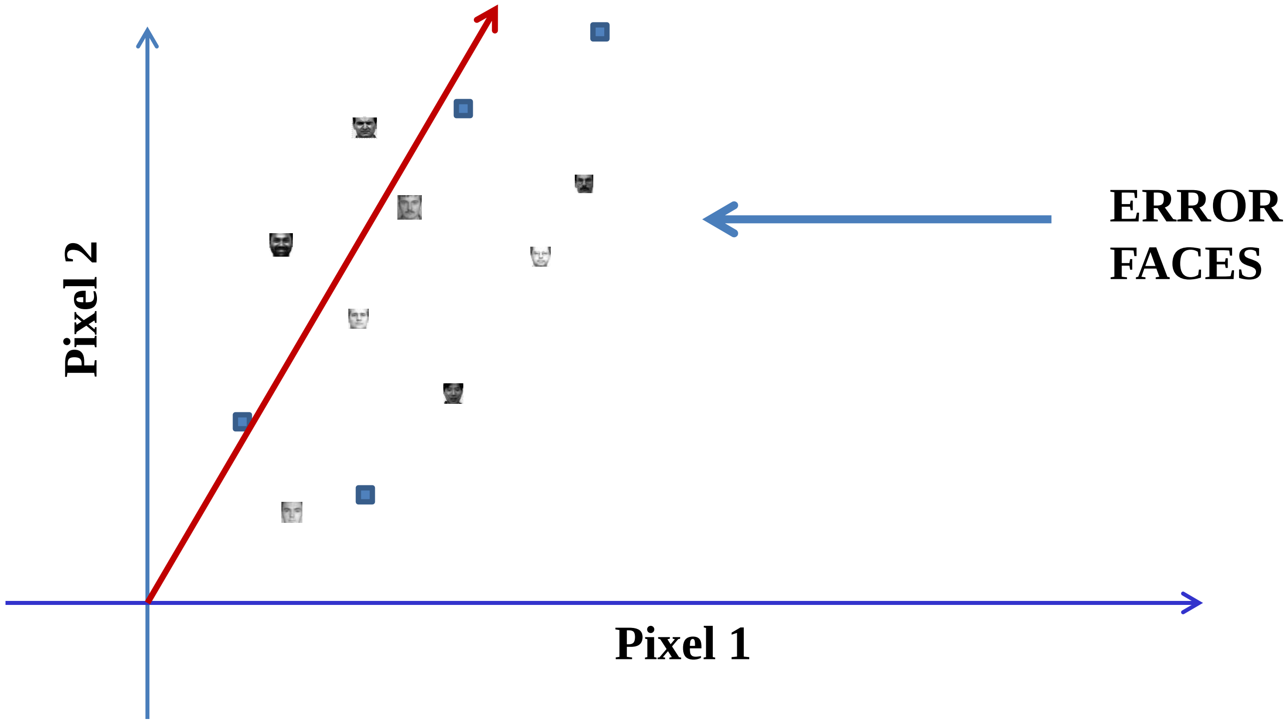


- Get the "error" faces by subtracting the first-level approximation from the original image

Solution: Iterate

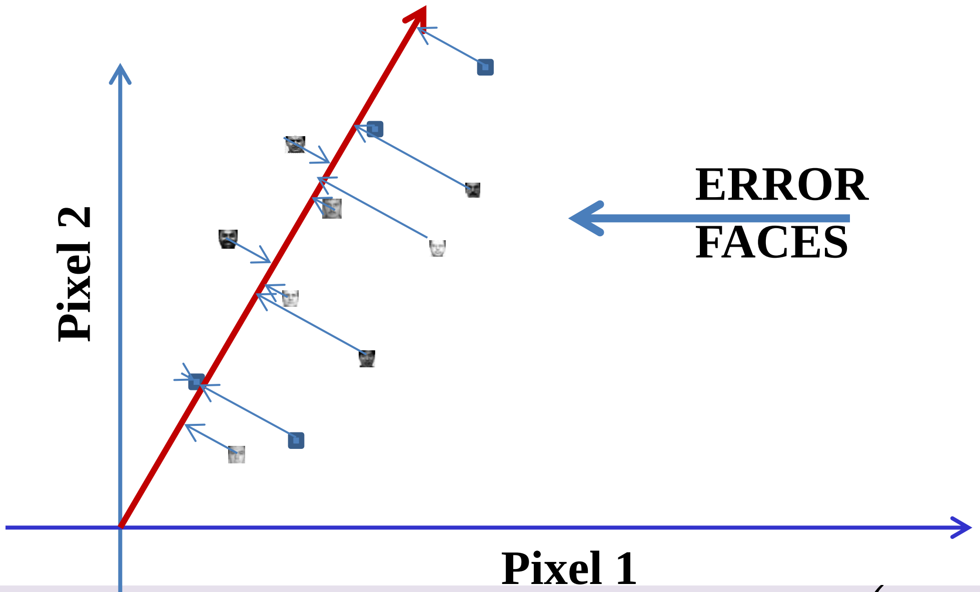
- Get the “error” faces by subtracting the first-level approximation from the original image
- Repeat the estimation on the “error” images

Abstracting the problem: Finding the *second* typical face



- Each "point" represents an *error* face in "pixel space"
- Find the vector V_2 such that the projection of these error faces on V_2 results in the least error

Minimizing error



The same math applies but now to the set of *error data points*

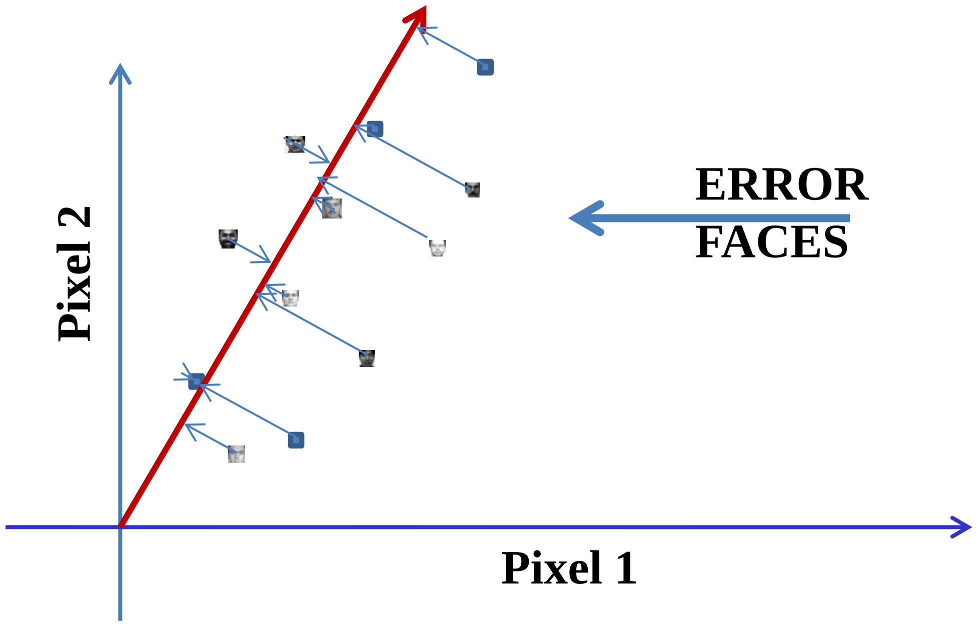
$$E = \sum_i \mathbf{e}_i^T \mathbf{e}_i - \sum_i \mathbf{e}_i^T \mathbf{w}^T \mathbf{e}_i + \lambda\left(\mathbf{v}^T \mathbf{v} - 1\right)$$

- Differentiating w.r.t $\mathbf{v}$ and equating to 0

$$-2\sum_i \mathbf{e}_i \mathbf{e}_i^T \mathbf{v} + 2\lambda \mathbf{v} = 0$$

$$\left(\sum_i \mathbf{e}_i \mathbf{e}_i^T\right)\mathbf{v} = \lambda \mathbf{v}$$

Minimizing error



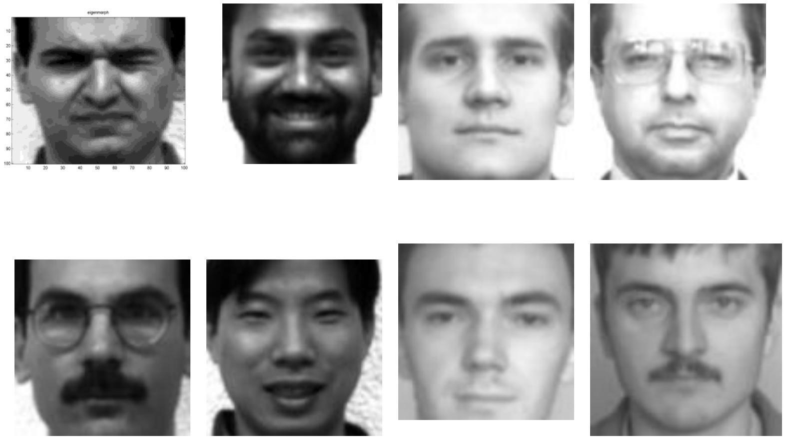
The same math applies but now to the set of *error data points*

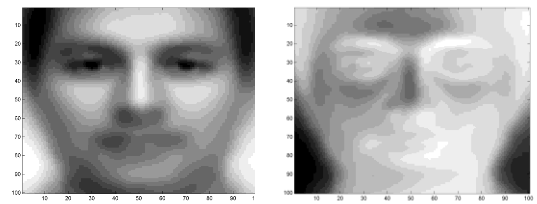
- The minimum-error basis is found by solving

$$\mathbf{R}_e \mathbf{v}_2 = \lambda \mathbf{v}_2 \qquad \mathbf{R}_e = \sum \mathbf{e}\mathbf{e}^T$$

- $\mathbf{v}_2$ is an Eigen vector of the correlation matrix $\mathbf{R}_e$ corresponding to the largest eigen value λ of $\mathbf{R}_e$

Which gives us our second typical face

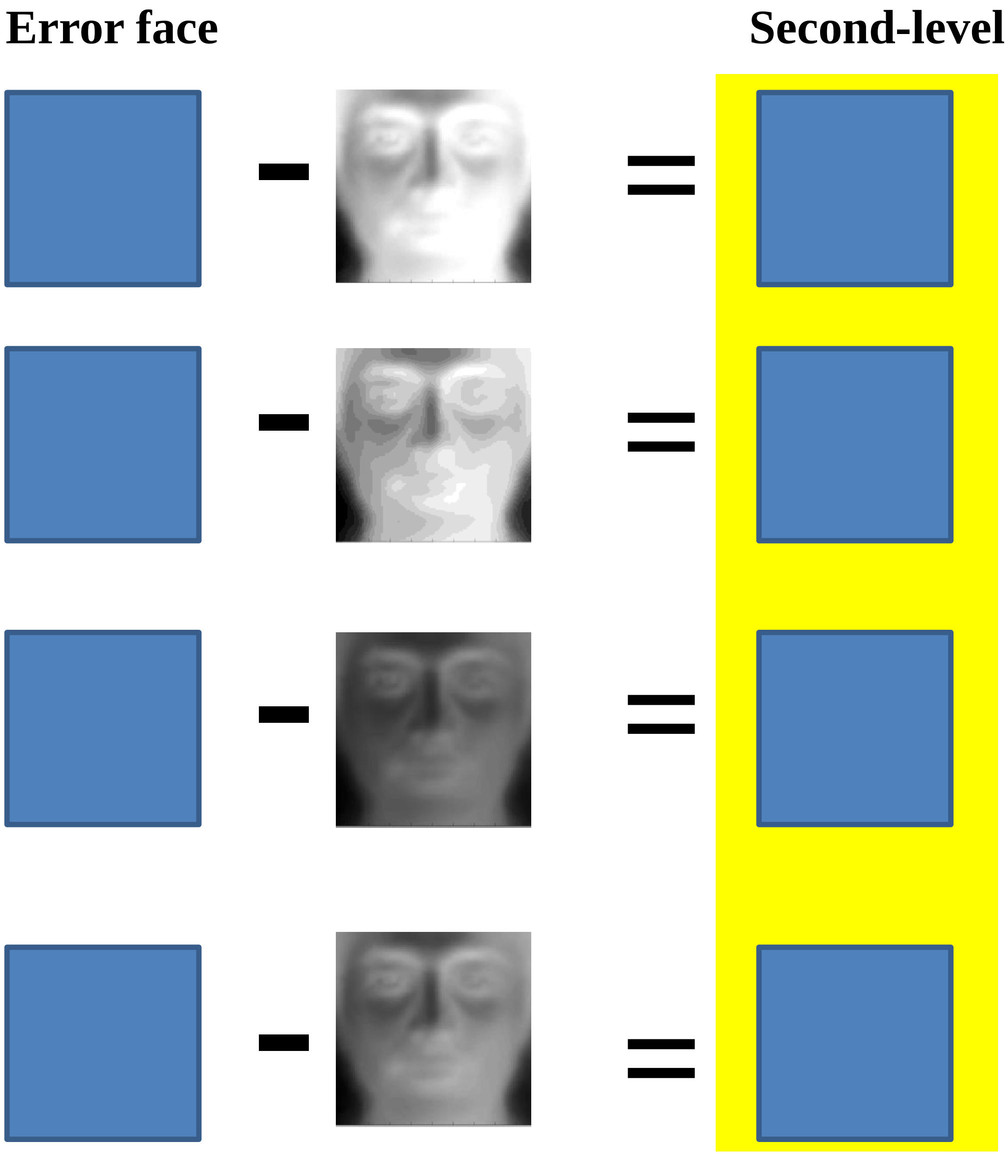




- But approximation with the two faces will *still* result in error
- So we need more typical faces to explain *this* error

- We can do this by subtracting the appropriately scaled version of the second “typical” face from the error images and repeating the process

Solution: Iterate



- Get the second-level "error" faces by subtracting the scaled second typical face from the first-level error
- Repeat the estimation on the second-level "error" images

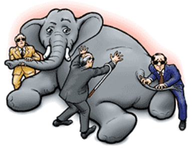
An interesting property

- Each "typical face" will be orthogonal to all other typical faces
  - Because each of them is learned to explain what the rest could not
  - None of these faces can explain one another!

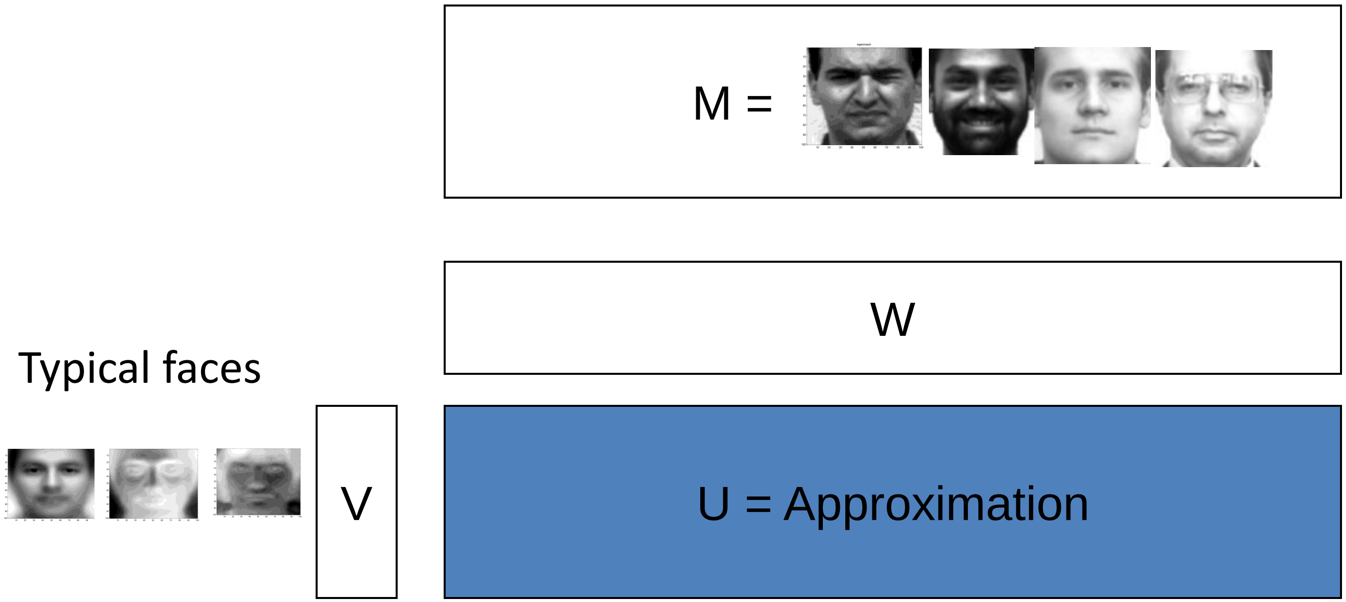
To add more faces

- We can continue the process, refining the error each time
  - An instance of a procedure is called “Gram-Schmidt” orthogonalization

- OR… we can do it all at once

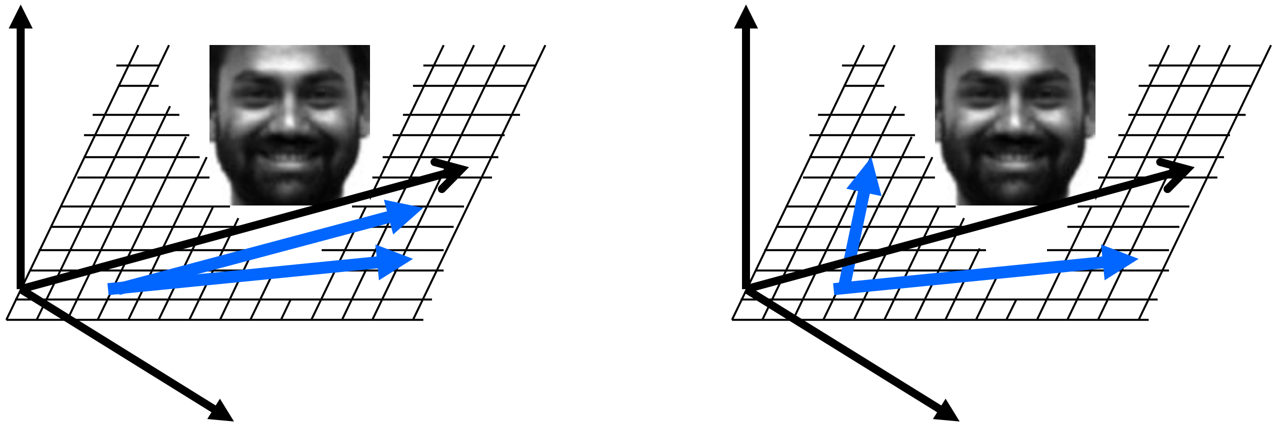


With many typical faces



- Approximate **every** face f as $f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k$
- Here W, V and U are ALL unknown and must be determined
  - Such that the squared error between U and M is minimum

With multiple bases

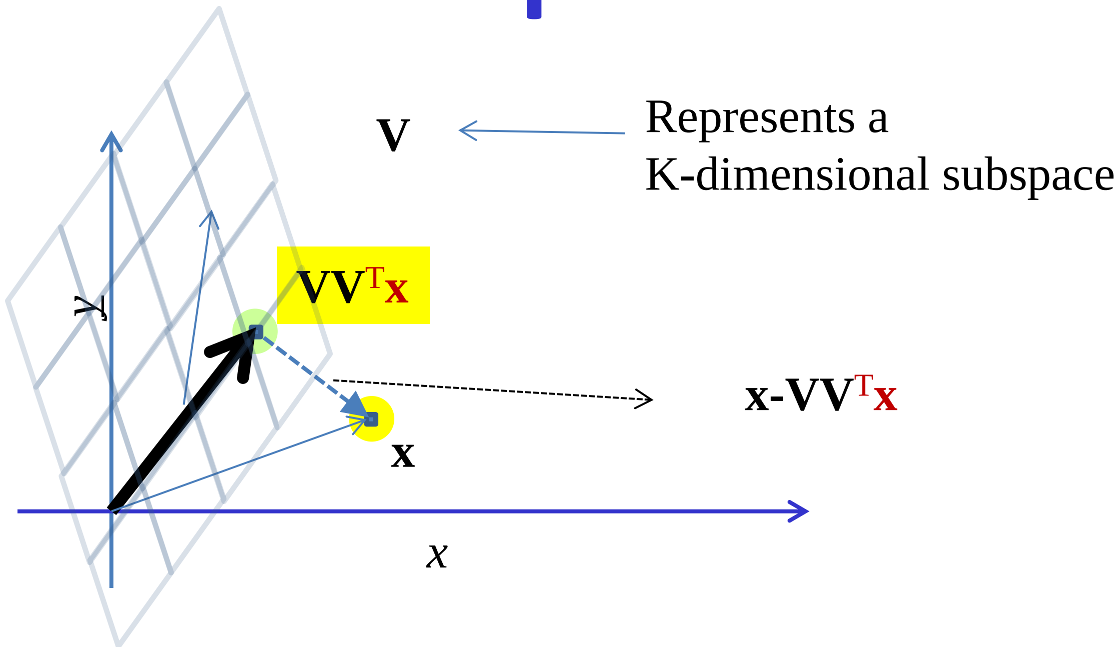


- **Assumption: all bases v_1 v_2 v_3.. are unit length**
- **Assumption: all bases are orthogonal to one another: $v_i^T v_j = 0$ if $i \neq j$**
  - We are trying to find the optimal K-dimensional subspace to project the data
  - Any set of vectors in this subspace will define the subspace
  - Constraining them to be orthogonal does not change this

- I.e. if $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \ldots]$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$
  - $\text{Pinv}(\mathbf{V}) = \mathbf{V}^T$

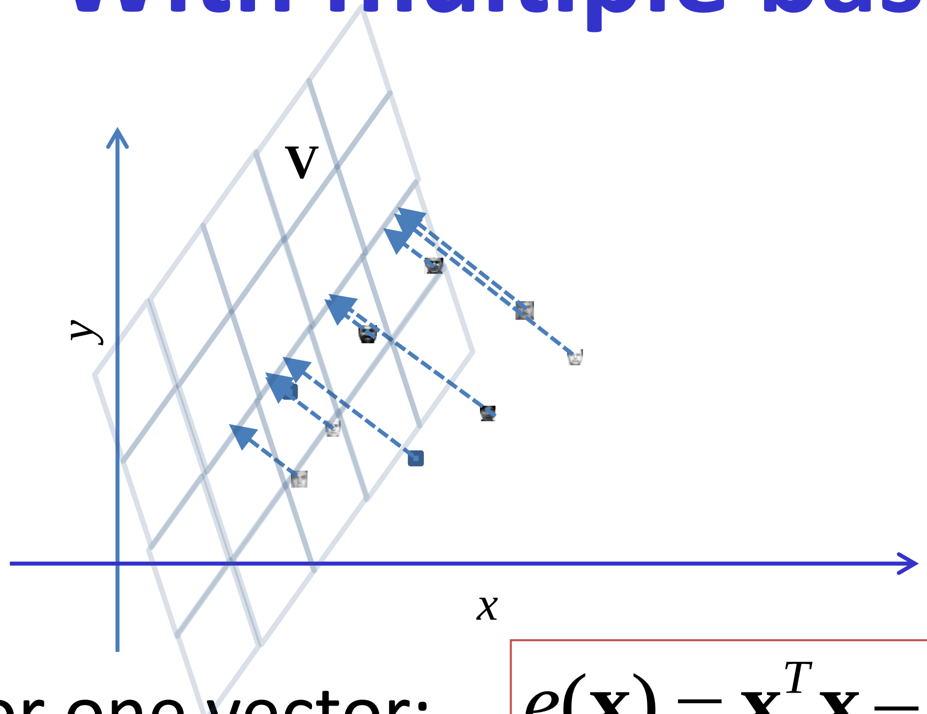
- Projection matrix for $\mathbf{V} = \mathbf{V}\text{Pinv}(\mathbf{V}) = \mathbf{V}\mathbf{V}^T$

With multiple bases



- Projection for a vector $\hat{\mathbf{x}} = \mathbf{V}\mathbf{V}^T\mathbf{x}$
- Error vector = $\mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \mathbf{V}\mathbf{V}^T\mathbf{x}$
- Error length = $e(\mathbf{x}) = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{V}\mathbf{V}^T\mathbf{x}$

With multiple bases



- Error for one vector: $e(\mathbf{x}) = \mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{V}\mathbf{V}^T\mathbf{x}$
- Error for many vectors

$$E = \sum_i \mathbf{x}_i^T\mathbf{x}_i - \sum_i \mathbf{x}_i^T\mathbf{V}\mathbf{V}^T\mathbf{x}_i$$

- **Goal: Estimate V to minimize this error!**

Minimizing error

- With constraint $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, objective to minimize

$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{V}\mathbf{V}^T \mathbf{x}_i + trace\left(\Lambda\left(\mathbf{V}^T\mathbf{V} - \mathbf{I}\right)\right)$$

  - Note: now Λ is a diagonal matrix
  - The constraint simply ensures that $\mathbf{v}^{\mathrm{T}}\mathbf{v} = 1$ for every basis

- Differentiating w.r.t $\mathbf{V}$ and equating to 0

$$-2\left(\sum_i \mathbf{x}_i \mathbf{x}_i^T\right)\mathbf{V} + 2\Lambda\mathbf{V} = 0$$

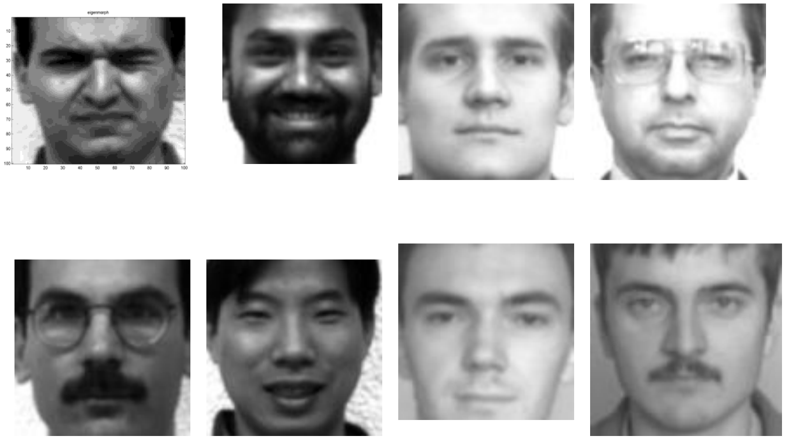
$$\mathbf{R}\mathbf{V} = \Lambda\mathbf{V}$$

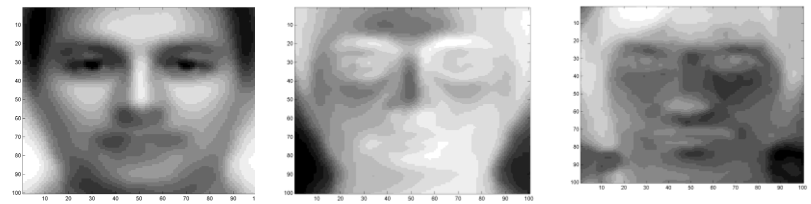
Finding the optimal K bases

$$\mathbf{RV} = \Lambda\mathbf{V}$$

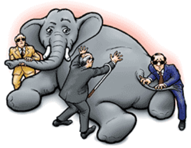
- Compute the Eigendecompsition of the correlation matrix
- Select *K* Eigen vectors
- But which *K*?
- Total error = $$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_{j=1}^{K} \lambda_j$$
- Select *K* eigen vectors corresponding to the *K* largest Eigen values

Eigen Faces!

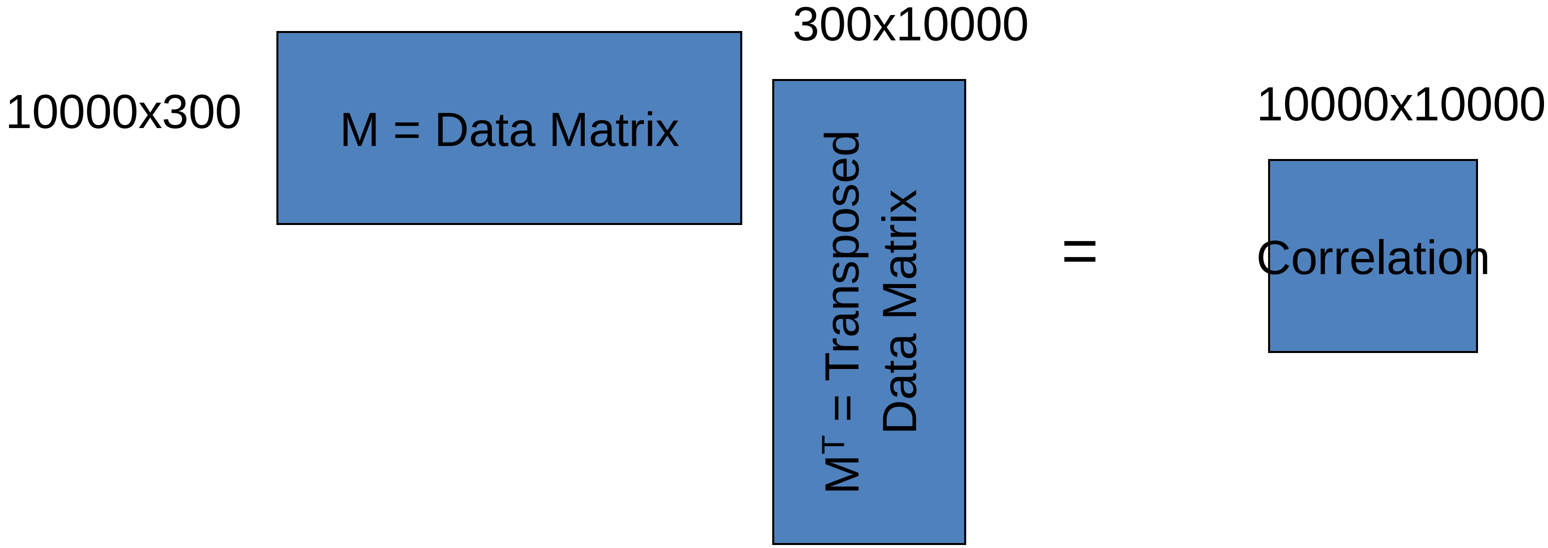




- Arrange your input data into a matrix $\mathbf{X}$
- Compute the correlation $\mathbf{R} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$
- Solve the Eigen decomposition: $\mathbf{R}\mathbf{V} = \Lambda\mathbf{V}$
- The Eigen vectors corresponding to the K largest eigen values are our optimal bases
- We will refer to these as *eigen faces.*



How many Eigen faces



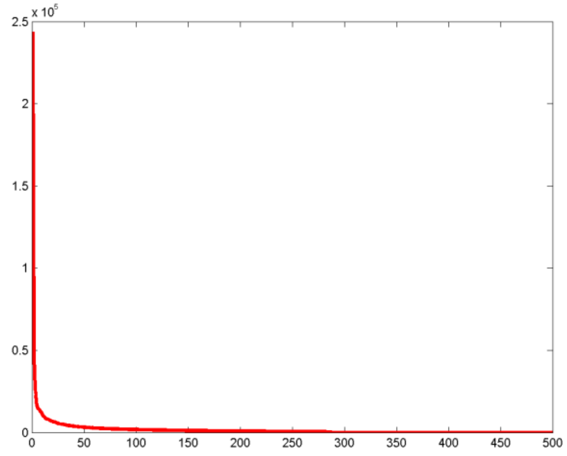
- How to choose "K" (number of Eigen faces)
- Lay all faces side by side in vector form to form a matrix
  – In my example: 300 faces. So the matrix is 10000 x 300
- Multiply the matrix by its transpose
  – The correlation matrix is 10000x10000

Eigen faces

[U,S] = eig(correlation)

$$S = \begin{bmatrix} \lambda_1 & . & 0 & . & 0 \\ 0 & \lambda_2 & 0 & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & 0 & . & \lambda_{10000} \end{bmatrix}$$

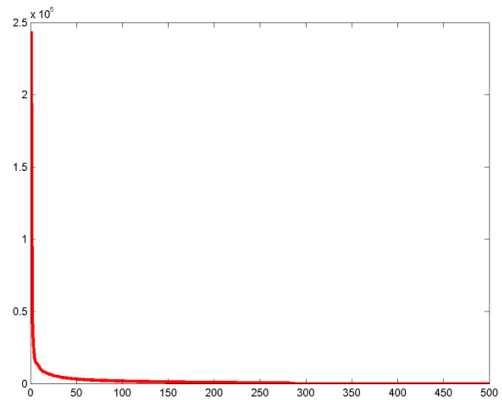
$$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \cdots \end{bmatrix}$$



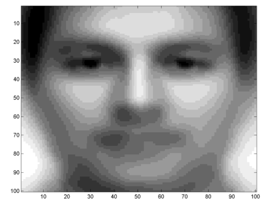
- Compute the eigen vectors
  - Only 300 of the 10000 eigen values are non-zero
    - Why?
- Retain eigen vectors with high eigen values (>0)
  - Could use a higher threshold

Eigen Faces

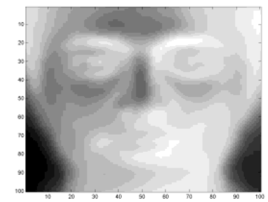
$$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \cdots \end{bmatrix}$$

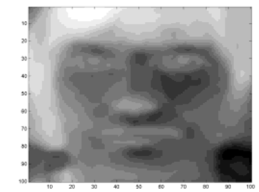


eigenface1



eigenface2

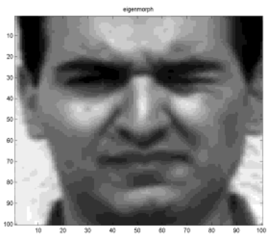
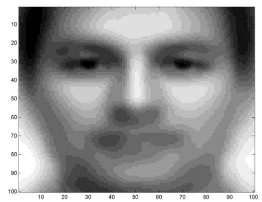
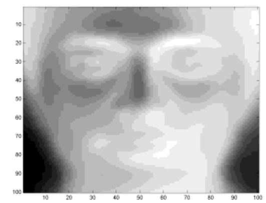
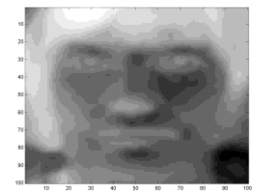


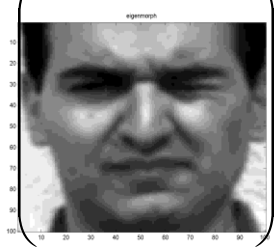


eigenface3

- The eigen vector with the highest eigen value is the first typical face
- The vector with the second highest eigen value is the second typical face.
- Etc.

Representing a face

 = w_1  + w_2  + w_3  •••

Representation $\left(\right.$  $\left.\right)$ = $[w_1\ w_2\ w_3\ \ldots.]^T$

- The weights with which the eigen faces must be combined to compose the face are used to represent the face!

Energy Compaction Example

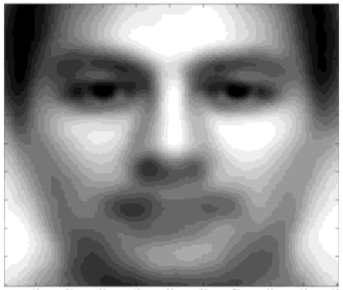
- One outcome of the "energy compaction principle": the approximations are recognizable

- Approximating a face with one basis:

$$f = w_1 \mathbf{v}_1$$

Energy Compaction Example

- One outcome of the "energy compaction principle": the approximations are recognizable

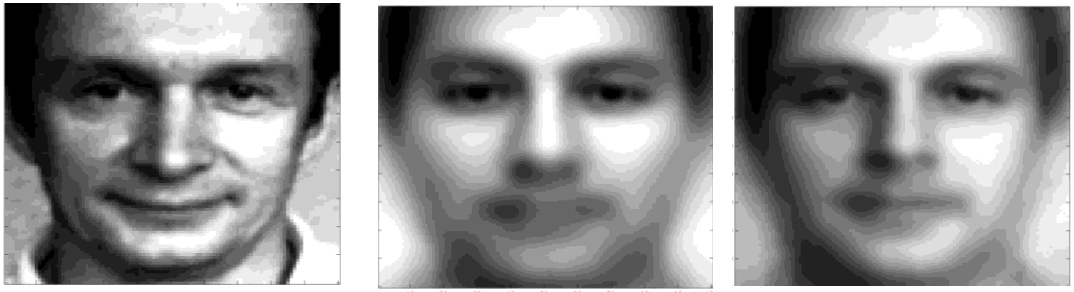


- Approximating a face with one Eigenface:

$$f = w_1 \mathbf{v}_1$$

Energy Compaction Example

- One outcome of the "energy compaction principle": the approximations are recognizable

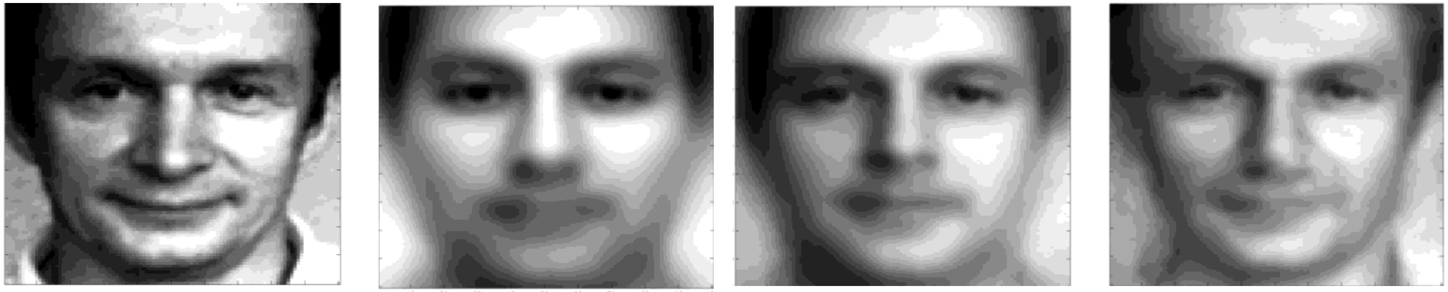


- Approximating a face with 10 eigenfaces:

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots w_{10}\mathbf{v}_{10}$$

Energy Compaction Example

- One outcome of the "energy compaction principle": the approximations are recognizable

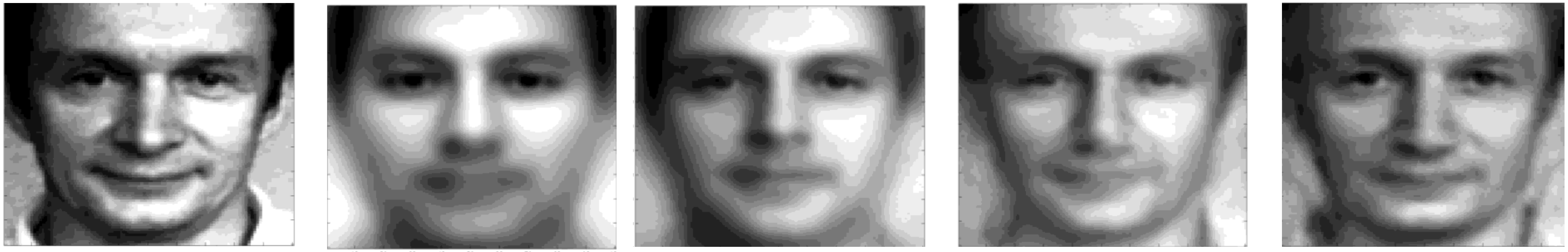


- Approximating a face with 30 eigenfaces:

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots + w_{10}\mathbf{v}_{10} + \ldots + w_{30}\mathbf{v}_{30}$$

Energy Compaction Example

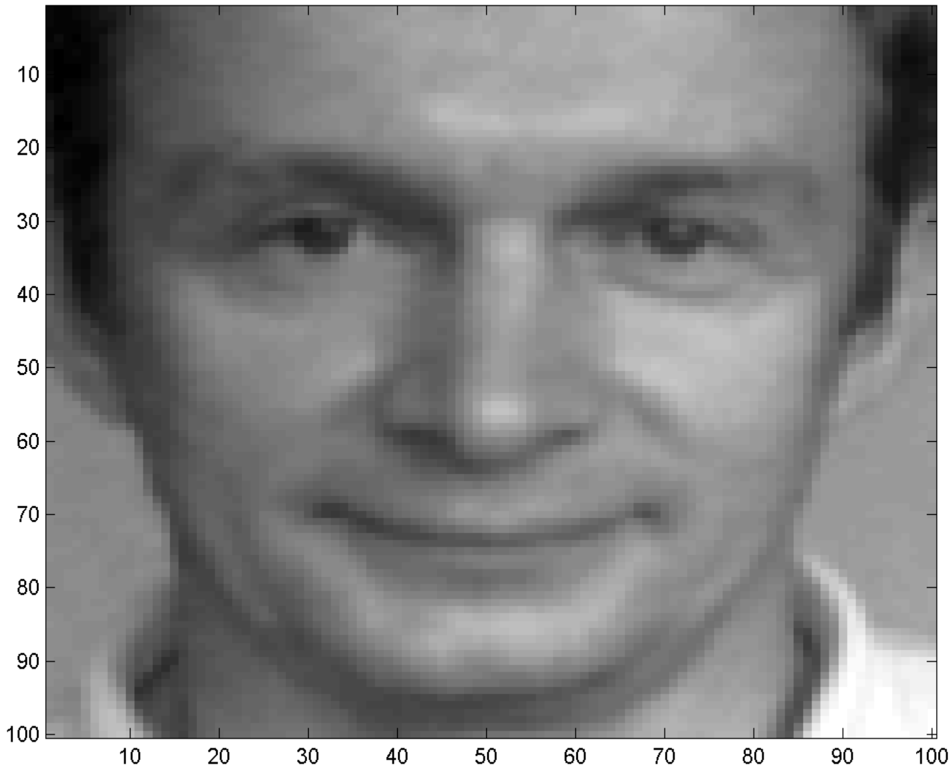
- One outcome of the "energy compaction principle": the approximations are recognizable



- Approximating a face with 60 eigenfaces:

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots + w_{10}\mathbf{v}_{10} + \ldots + w_{30}\mathbf{v}_{30} + \ldots + w_{60}\mathbf{v}_{60}$$

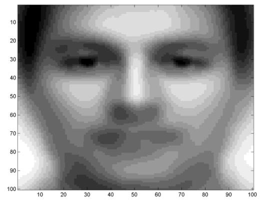
How did I do this?



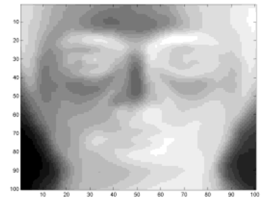
- Hint: only changing weights assigned to Eigen faces..

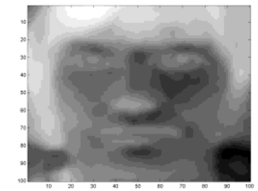
Class specificity

eigenface1



eigenface2





eigenface3

- The Eigenimages (bases) are very specific to the class of data they are trained on
  - Faces here
- They will not be useful for other classes

Class specificity

- Eigen bases are class specific

- Composing a fishbowl from Eigenfaces

Class specificity

- Eigen bases are class specific

- Composing a fishbowl from Eigenfaces
- With 1 basis

$$f = w_1 \mathbf{v}_1$$

Class specificity

- Eigen bases are class specific



- Composing a fishbowl from Eigenfaces
- With 10 bases

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots + w_{10}\mathbf{v}_{10}$$

Class specificity

- Eigen bases are class specific




- Composing a fishbowl from Eigenfaces
- With 30 bases

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots + w_{10}\mathbf{v}_{10} + \ldots + w_{30}\mathbf{v}_{30}$$

Class specificity

- Eigen bases are class specific



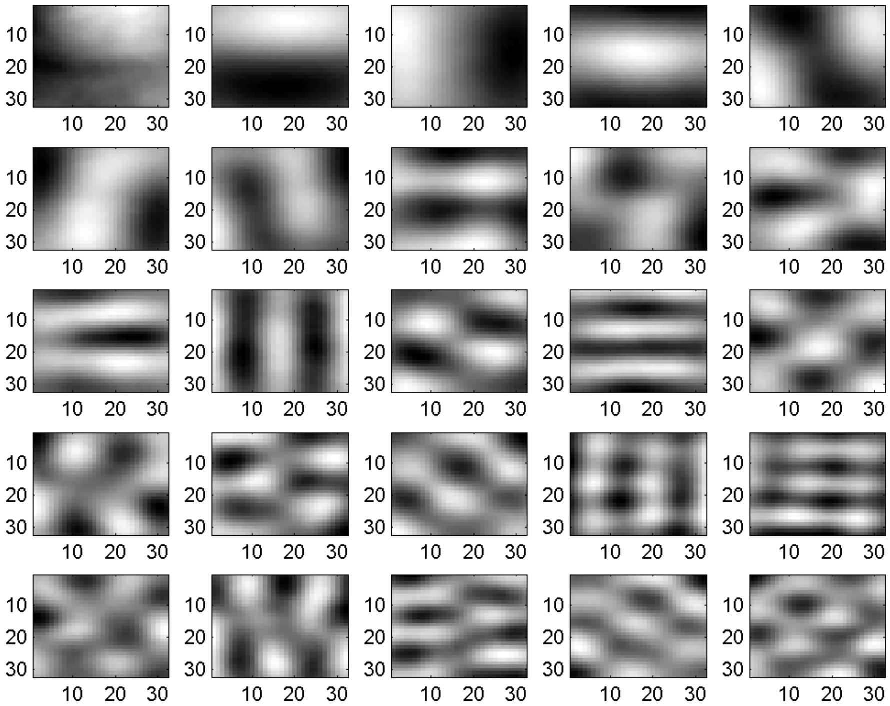


- Composing a fishbowl from Eigenfaces
- With 100 bases

$$f = w_1\mathbf{v}_1 + w_2\mathbf{v}_2 + \ldots + w_{10}\mathbf{v}_{10} + \ldots + w_{30}\mathbf{v}_{30} + \ldots + w_{100}\mathbf{v}_{100}$$

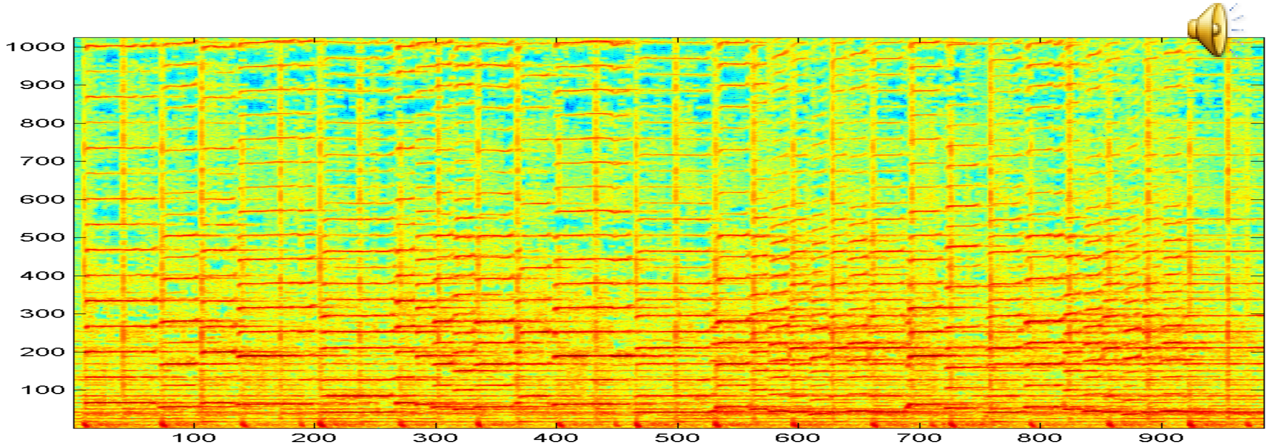
Universal bases

- Universal bases..



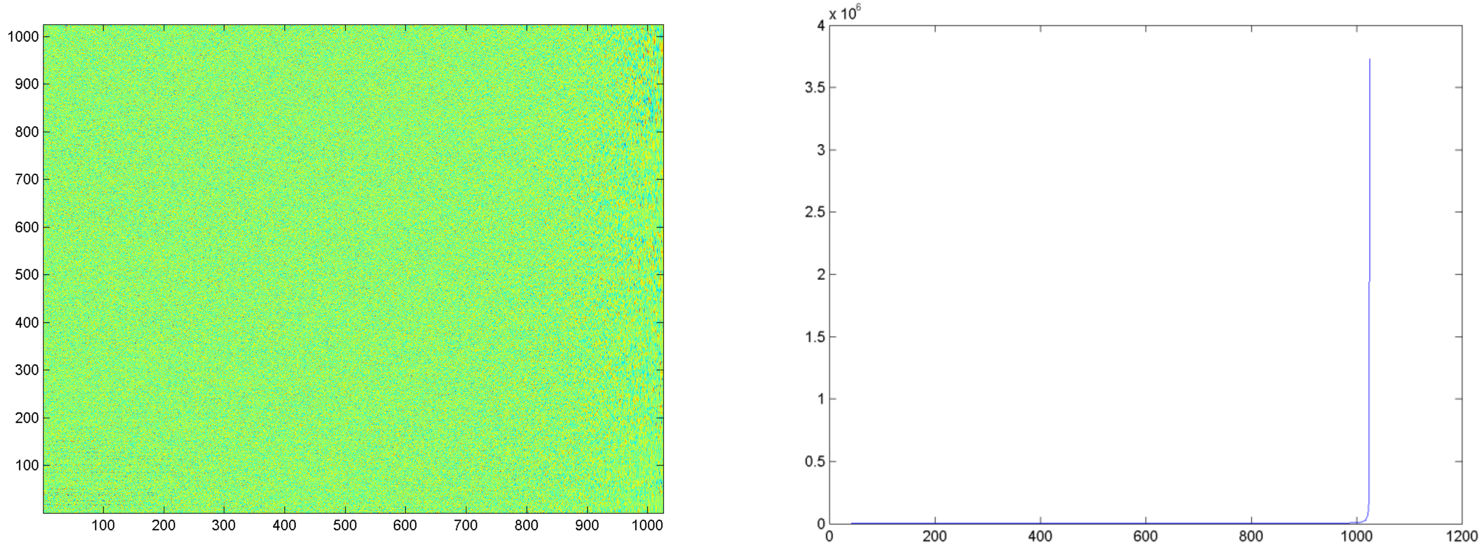
- End up looking a lot like *discrete cosine transforms!!!!*
- *DCTs are the best "universal" bases*
  - *If you don't know what your data are, use the DCT*

An audio example



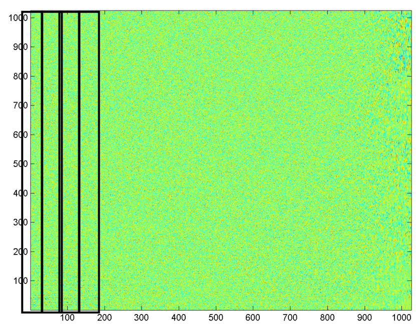
- The spectrogram has 974 vectors of dimension 1025
- The covariance matrix is size 1025 x 1025
- There are 1025 eigenvectors

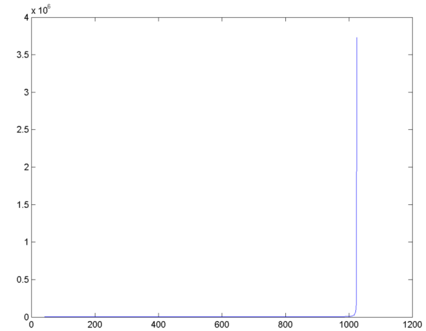
Eigenvalues and Eigenvectors

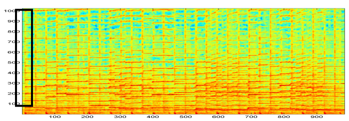


- Left panel: Matrix with 1025 eigen vectors
- Right panel: Corresponding eigen values
  - Most Eigen values are close to zero
    - The corresponding eigenvectors are "unimportant"

Eigenvalues and Eigenvectors





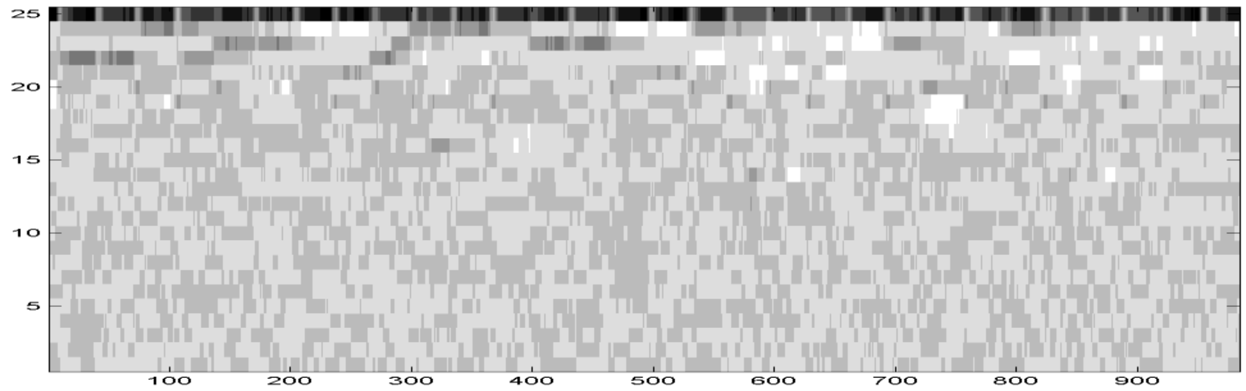


Vec = a1 *eigenvec1 + a2 * eigenvec2 + a3 * eigenvec3 …

- The vectors in the spectrogram are linear combinations of all 1025 Eigen vectors
- The Eigen vectors with low Eigen values contribute very little
  - The average value of a_i is proportional to the square root of the Eigenvalue
  - Ignoring these will not affect the composition of the spectrogram

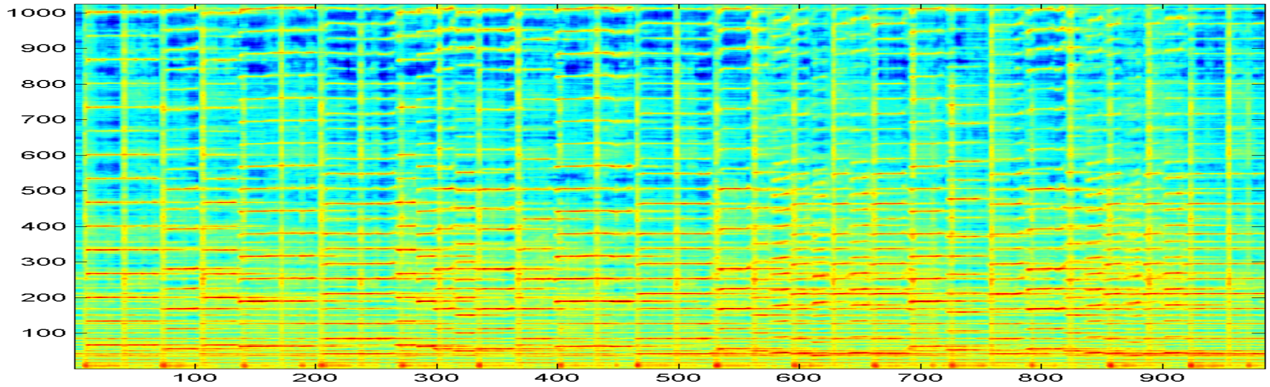
An audio example

$$V_{reduced} = [V_1 \quad . \quad . \quad V_{25}]$$
$$M_{lowdim} = Pinv(V_{reduced})M$$



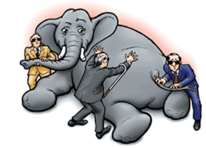
- The same spectrogram projected down to the 25 eigen vectors with the highest eigen values
  - Only the 25-dimensional weights are shown
    - The weights with which the 25 eigen vectors must be added to compose a least squares approximation to the spectrogram

An audio example

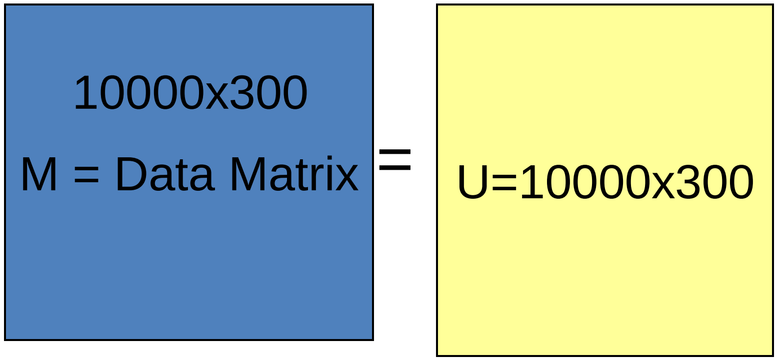


$$M_{reconstructed} = V_{reduced} M_{lowdim}$$

- The same spectrogram constructed from only the 25 Eigen vectors with the highest Eigen values
  - Looks similar
    - With 100 Eigenvectors, it would be indistinguishable from the original
  - Sounds pretty close
  - But now sufficient to store 25 numbers per vector (instead of 1024)



SVD instead of Eigen



$$U = \begin{bmatrix} \text{eigenface1} & \text{eigenface2} & \cdots \end{bmatrix}$$

- Do we need to compute a 10000 x 10000 correlation matrix and then perform Eigen analysis?
  - Will take a very long time on your laptop
- SVD
  - Only need to perform "Thin" SVD. Very fast
    - U = 10000 x 300
      - The columns of U are the eigen faces!
      - The Us corresponding to the "zero" eigen values are not computed
    - S = 300 x 300
    - V = 300 x 300

Using SVD to compute Eigenbases

[U, S, V] = SVD(X)

- U will have the Eigenvectors
- Thin SVD for 100 bases:

[U,S,V] = svds(X, 100)

- Much more efficient

Eigen Decomposition of data

- Nothing magical about faces or sound – can be applied to any data.
  - Eigen analysis is one of the key components of data compression and representation
  - Represent N-dimensional data by the weights of the K leading Eigen vectors
    - Reduces effective dimension of the data from N to K
    - But requires knowledge of Eigen vectors

Eigen decomposition of what?

- Eigen decomposition of the *correlation* matrix

- Is there an alternate way?

Linear vs. Affine

- The model we saw
  - Approximate **every** face f as
    $$f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k$$
  - Linear combination of bases

- If you add a constant
  $$f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k + m$$
  - *Affine* combination of bases

Estimation with the constant

- Estimate

$$f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k + m$$

- Lets do this incrementally first:
- $f \approx m$
  - For every face
  - Find *m* to optimize the approximation

Estimation with the constant

- Estimate

  $$\mathrm{f} \approx \mathrm{m}$$

  – for *every f!*
- Error over all faces $E = \sum_f ||f - m||^2$
- Minimizing the error with respect to *m*, we simply get
  – $m = \frac{1}{N}\sum_f f$
- The *mean* of the data

Estimation the remaining

- Same procedure as before:
  - Remaining "typical faces" must model what the constant *m* could not
- Subtract the constant from every data point
  - $\hat{f} = f - m$
- Now apply the model:
  - $\hat{f} = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k$
- This is just Eigen analysis of the "mean-normalized" data
  - Also called the "centered" data

Estimating the Affine model

$$f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k + m$$

- First estimate the mean *m*

$$m = \frac{1}{N}\sum_f f$$

- Compute the correlation matrix of the "centered" data $\hat{f} = f - m$
  - $\boldsymbol{C} = \sum_f \hat{f}\hat{f}^T = \sum_f (f - m)(f - m)^T$
  - This is the *covariance* matrix of the set of f

Estimating the Affine model

$$f = w_{f,1} V_1 + w_{f,2} V_2 + \ldots + w_{f,k} V_k + m$$

- First estimate the mean m

$$m = \frac{1}{N}\sum_{f} f$$

- Compute the covariance matrix
  - $\boldsymbol{C} = \sum_f (f - m)(f - m)^T$

- Eigen decompose!

$$\mathbf{CV} = \Lambda\mathbf{V}$$

- The Eigen vectors corresponding to the top *k* Eigen values give us the bases V_k

Properties of the affine model

- The bases $V_1, V_2, \ldots, V_k$ are all orthogonal to one another
  - Eigen vectors of the symmetric Covariance matrix
- But they are *not* orthogonal to *m*
  - Because *m* is an unscaled constant

- We could jointly estimate all $V_1, V_2, \ldots, V_k$ and *m* by minimizing

$$\sum_f ||f - (\sum_f w_{f,i} V_i + m)||^2 + trace(\Lambda(\boldsymbol{V}^T\boldsymbol{V} - \boldsymbol{I}))$$

Linear vs. Affine

- The model we saw
  - Approximate **every** face f as
    $f = w_{f,1} V_1 + w_{f,2} V_2 + \dots + w_{f,k} V_k$
  - The ***Karhunen Loeve Transform***
  - Retains maximum ***Energy*** for any order k

- If you add a constant

$$f = w_{f,1} V_1 + w_{f,2} V_2 + \dots + w_{f,k} V_k + m$$

  - ***Principal Component Analysis***
  - Retains maximum ***Variance*** for any order k

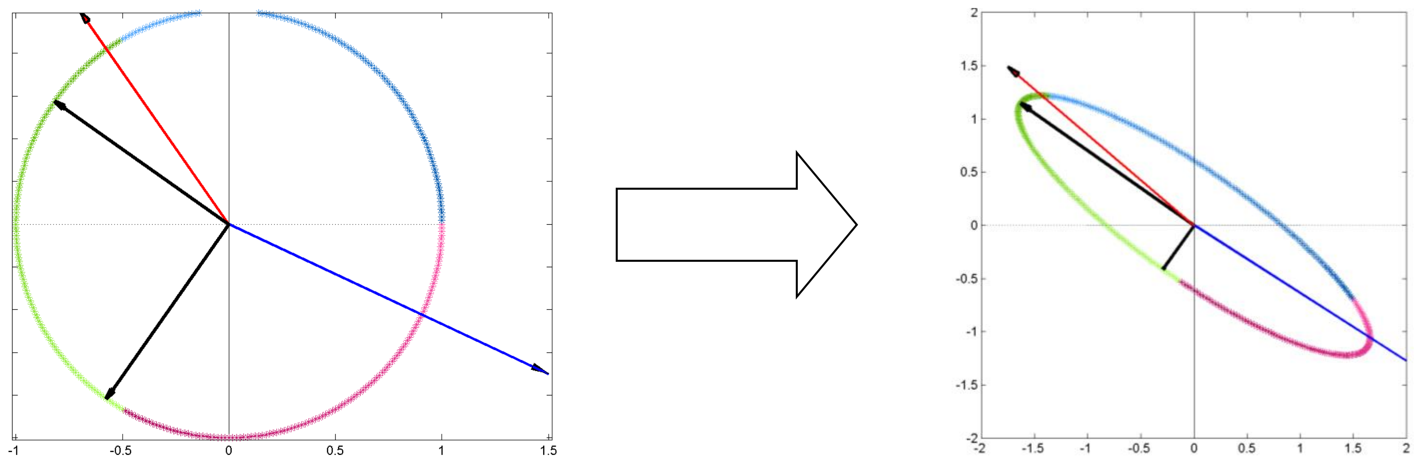
How do they relate

- Relationship between correlation matrix and covariance matrix

$$\mathbf{R} = \mathbf{C} + mm^{\mathrm{T}}$$

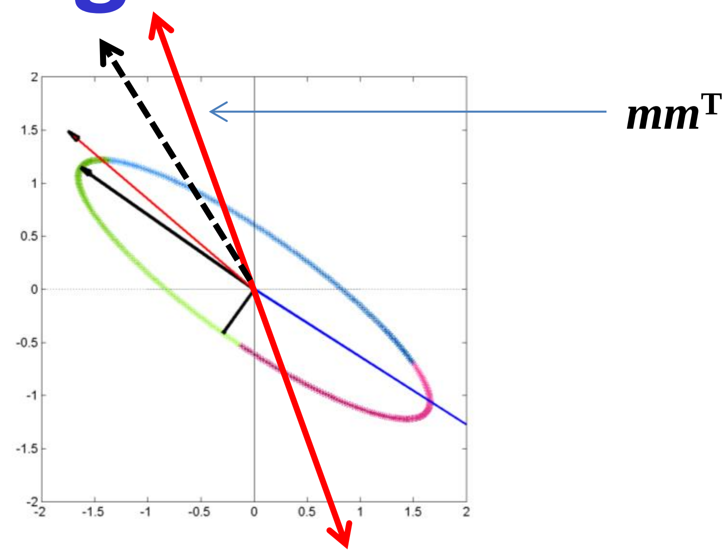
- *Karhunen Loeve* bases are Eigen vectors of **R**
- *PCA* bases are Eigen vectors of **C**
- How do they relate
  – Not easy to say..

The Eigen vectors



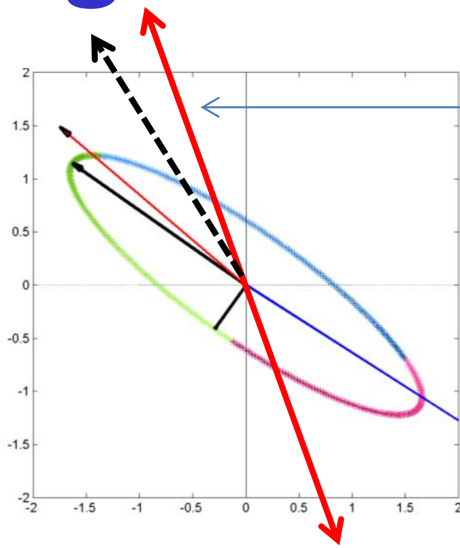
- The Eigen vectors of ***C*** are the major axes of the ellipsoid ***Cv***, where ***v*** are the vectors on the unit sphere

The Eigen vectors



- The Eigen vectors of ***R*** are the major axes of the ellipsoid ***Cv*** + $\boldsymbol{mm}^T\boldsymbol{v}$
- Note that $\boldsymbol{mm}^T$ has rank 1 and $\boldsymbol{mm}^T\boldsymbol{v}$ is a line

The Eigen vectors



- The principal Eigenvector of ***R*** lies between the principal Eigen vector of ***C*** and ***m***

$$\mathbf{e}_R = \alpha \mathbf{e}_C + (1-\alpha)\frac{\mathbf{m}}{\|\mathbf{m}\|}$$

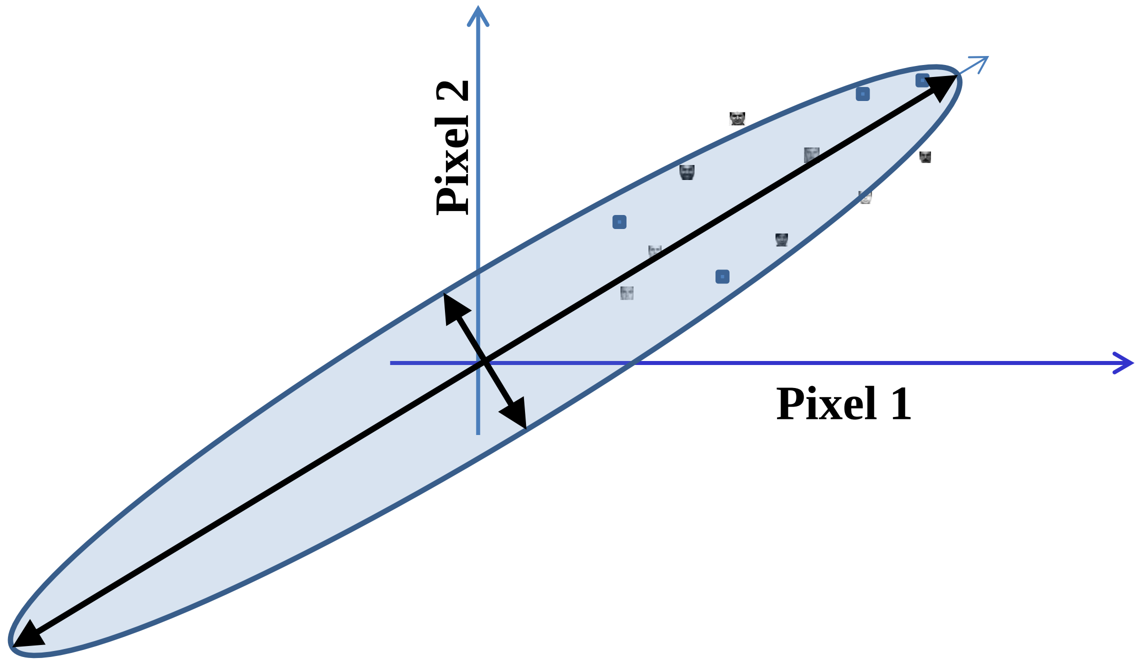
$$0 \le \alpha \le 1$$

- Similarly the principal Eigen *value*

$$\lambda_R = \alpha \lambda_C + (1-\alpha)\|\mathbf{m}\|^2$$

- Similar logic is not easily extendable to the other Eigenvectors, however

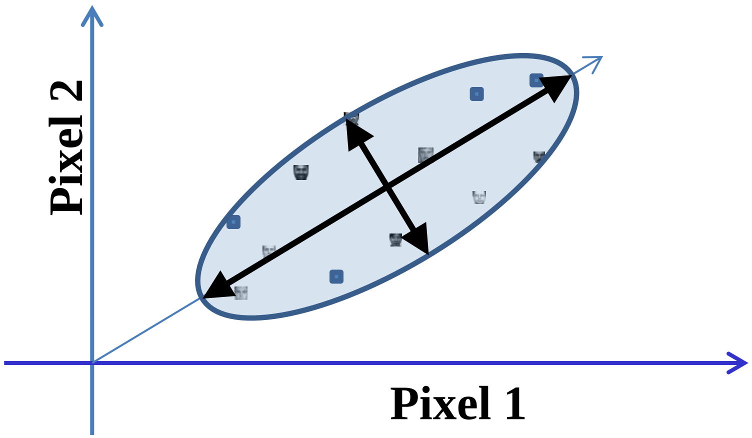
Eigenvectors



- Turns out: Eigenvectors of the *correlation* matrix represent the major and minor axes of an ellipse centered at the origin which encloses the data most compactly

- The SVD of data matrix X uncovers these vectors
  - **KLT**

Eigenvectors



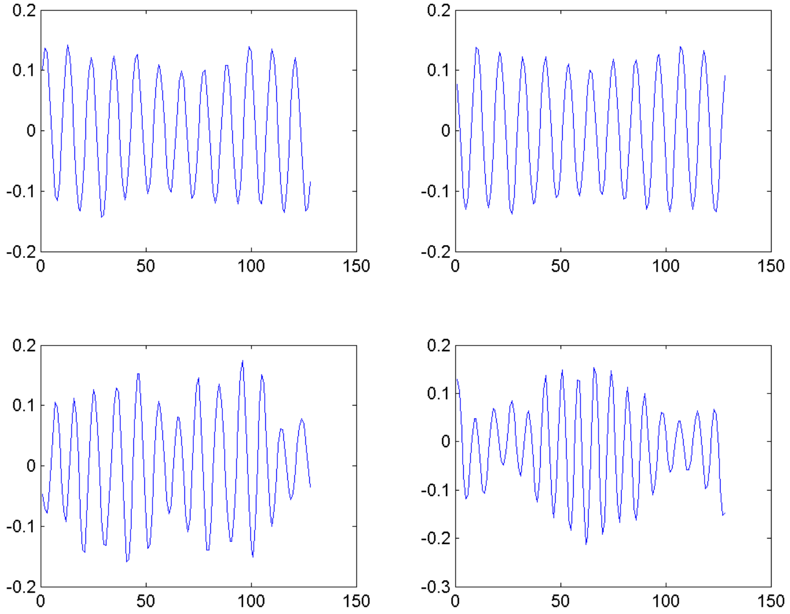
- Turns out: Eigenvectors of the *covariance* represent the major and minor axes of an ellipse centered at the *mean* which encloses the data most compactly

- PCA uncovers these vectors
- In practice, “Eigen faces” refers to *PCA* faces, and not KLT faces

What about sound?

- Finding Eigen bases for speech signals:

- Look like DFT/DCT
- Or wavelets



- DFTs are pretty good most of the time

Eigen Analysis

- Can often find surprising features in your data
- Trends, relationships, more
- Commonly used in recommender systems

- An interesting example..

Eigen Analysis

Figure1. Experiment setup @Wean Hall mechanical space. Pipe with arrow indicates a 10" diameter hot water pipe carrying pressurized hot water flow, on which piezoelectric sensors are installed every 10 ft. A National instruments data acquisition system is used to acquire and store the data for later processing.

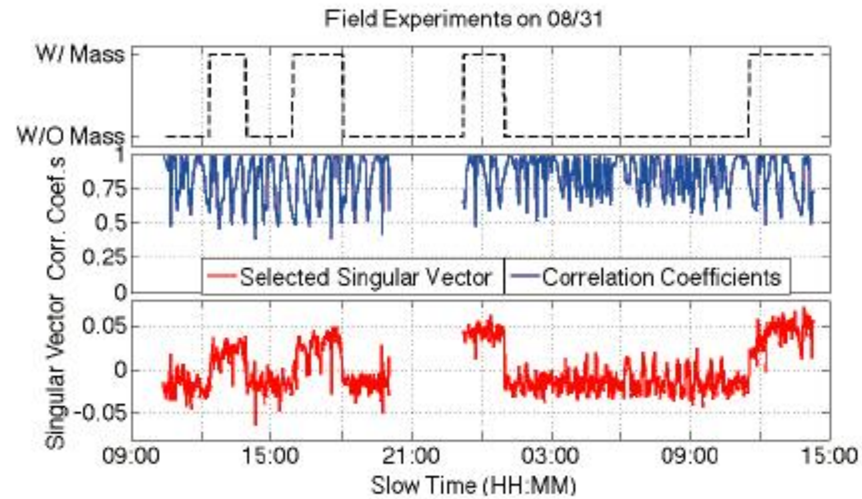


Figure 2. Damage detection results compared with conventional methods. **Top**: Ground truth of whether the pipe is damaged or not. **Middle**: Conventional method only captures temperature variations, and shows no indication of the presence of damage. **Bottom**: The SVD method clearly picks up the steps where damage are introduced and removed.

- Cheng Liu's research on pipes..
- SVD automatically separates useful and uninformative features