

Machine Learning for Signal Processing

Data driven representations:

1. Eigenrepresentations

Instructor: Bhiksha Raj

Linear Algebra recall

- What is $\mathbf{x}^T \mathbf{y}$
 - When \mathbf{y} is unit length

Linear Algebra recall

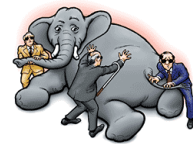
- What is $\mathbf{x}^T \mathbf{y}$
 - When \mathbf{y} is unit length
- What is the projection of \mathbf{x} onto \mathbf{y}
 - When \mathbf{y} is unit length

Linear Algebra recall

- What is $\mathbf{x}^T \mathbf{y}$
 - When \mathbf{y} is unit length
- What is the projection of \mathbf{x} onto \mathbf{y}
 - When \mathbf{y} is unit length
- What is the projection of \mathbf{x} onto $\mathbf{Y} = [\mathbf{y}_1 \mathbf{y}_1 \dots \mathbf{y}_K]$
 - When \mathbf{Y} is an orthogonal matrix

- **On with the topic for today...**

Recall: Representing images



aboard Apollo space capsule.
1038 x 1280 - 142k
LIFE



Apollo Xi
1280 x 1255 - 226k
LIFE



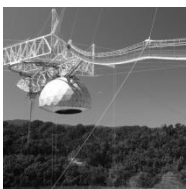
aboard Apollo space capsule.
1029 x 1280 - 128k
LIFE



Building Apollo space ship.
1280 x 1257 - 114k
LIFE



aboard Apollo space capsule.
1017 x 1280 - 130k
LIFE



Apollo Xi
1228 x 1280 - 181k
LIFE



Apollo 10 space ship, w.
1280 x 853 - 72k
LIFE



Splashdown of Apollo XI mission.
1280 x 866 - 184k
LIFE



Earth seen from space during the
1280 x 839 - 60k
LIFE



Apollo Xi
844 x 1280 - 123k
LIFE



Apollo 8
1278 x 1280 - 74k
LIFE



working on Apollo space project.
1223 x 956 - 117k
LIFE



the moon as seen from Apollo 8
1223 x 1280 - 214k
LIFE



Apollo 11
1280 x 1277 - 142k
LIFE

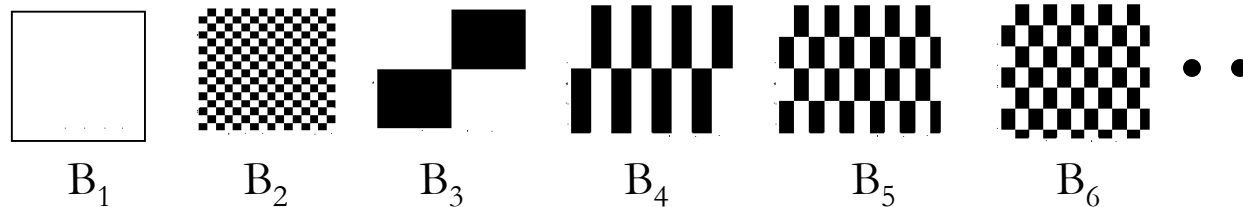


Apollo 8 Crew
968 x 1280 - 125k
LIFE

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



Adding more bases



- Checkerboards with different variations

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

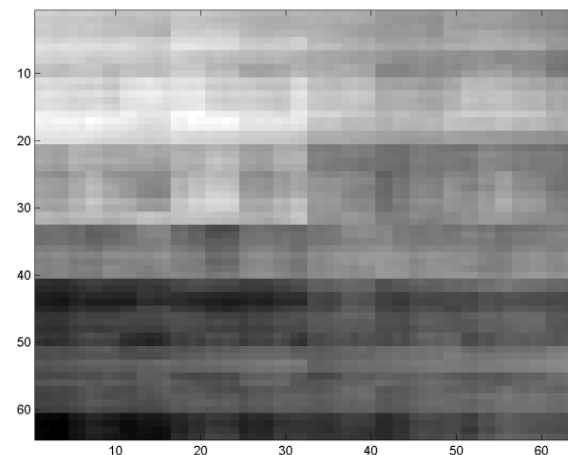
$$W = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ \vdots \end{bmatrix}$$

$$B = [B_1 \ B_2 \ B_3]$$

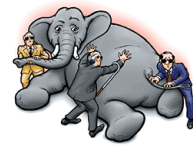
$$BW \approx \text{Image}$$

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

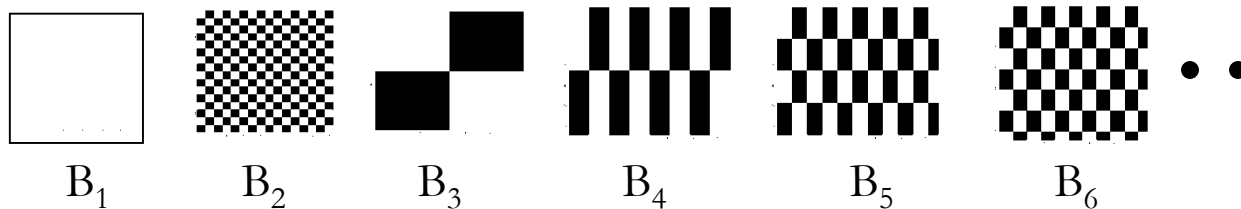
$$\text{PROJECTION} = BW$$



Getting closer at 625 bases!

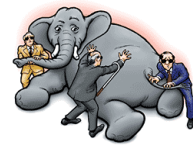


“Bases”



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

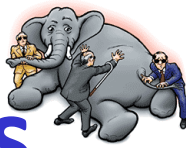
- “Bases” are the “standard” units such that all instances can be expressed a weighted combinations of these units
- Ideal requirements: Bases must be orthogonal
- Checkerboards are one choice of bases
 - Orthogonal
 - But not “smooth”
- Other choices of bases: Complex exponentials, Wavelets, etc..



Data specific bases?

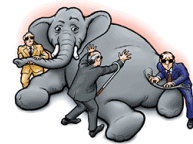
- **Issue: The bases we have considered so far are *data agnostic***
 - Checkerboards, Complex exponentials, Wavelets..
 - We use the same bases regardless of the data we analyze
 - Image of face vs. Image of a forest
 - Segment of speech vs. Seismic rumble
- How about data specific bases
 - Bases that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

Data-specific description of faces

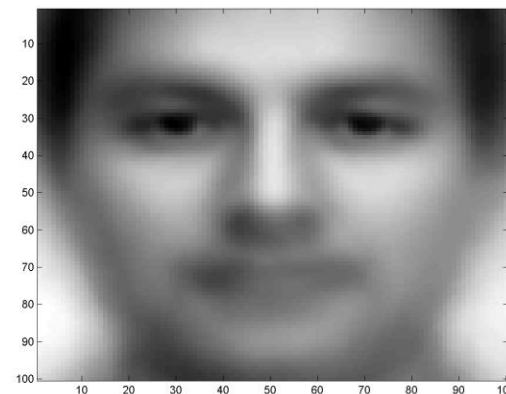


- A collection of images
 - All normalized to 100x100 pixels
- What is common among all of them?
 - Do we have a common descriptor?

A typical face



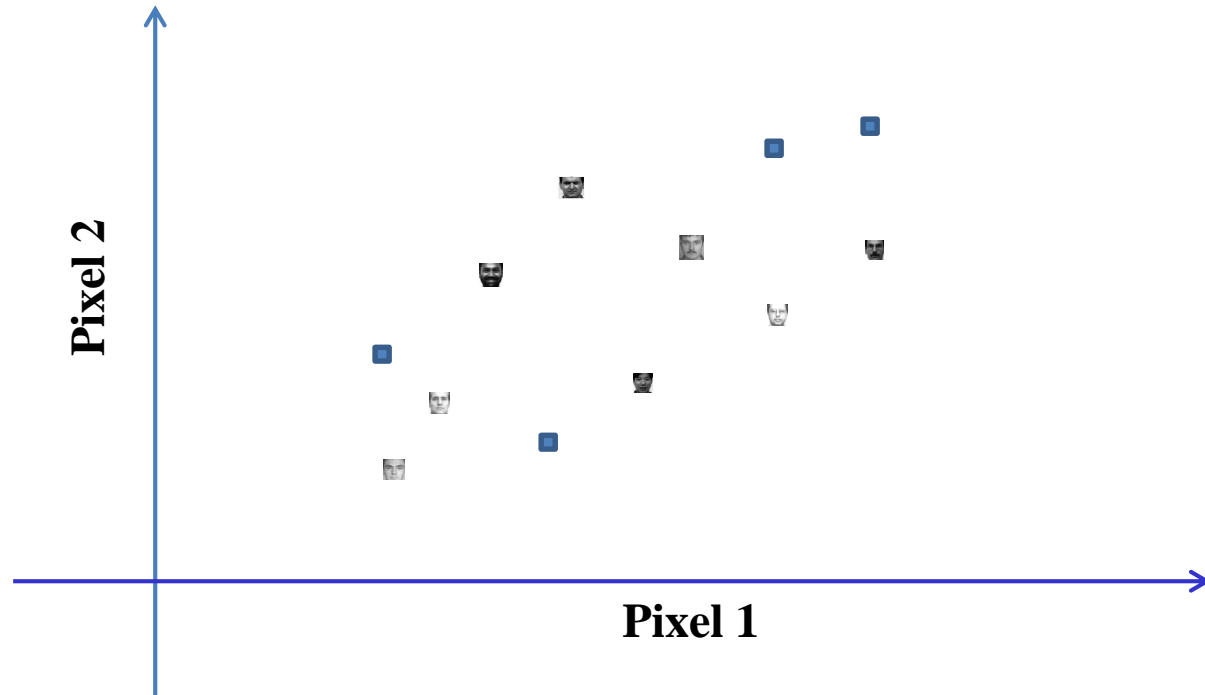
The typical face



- **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 ?

Abstracting the problem:

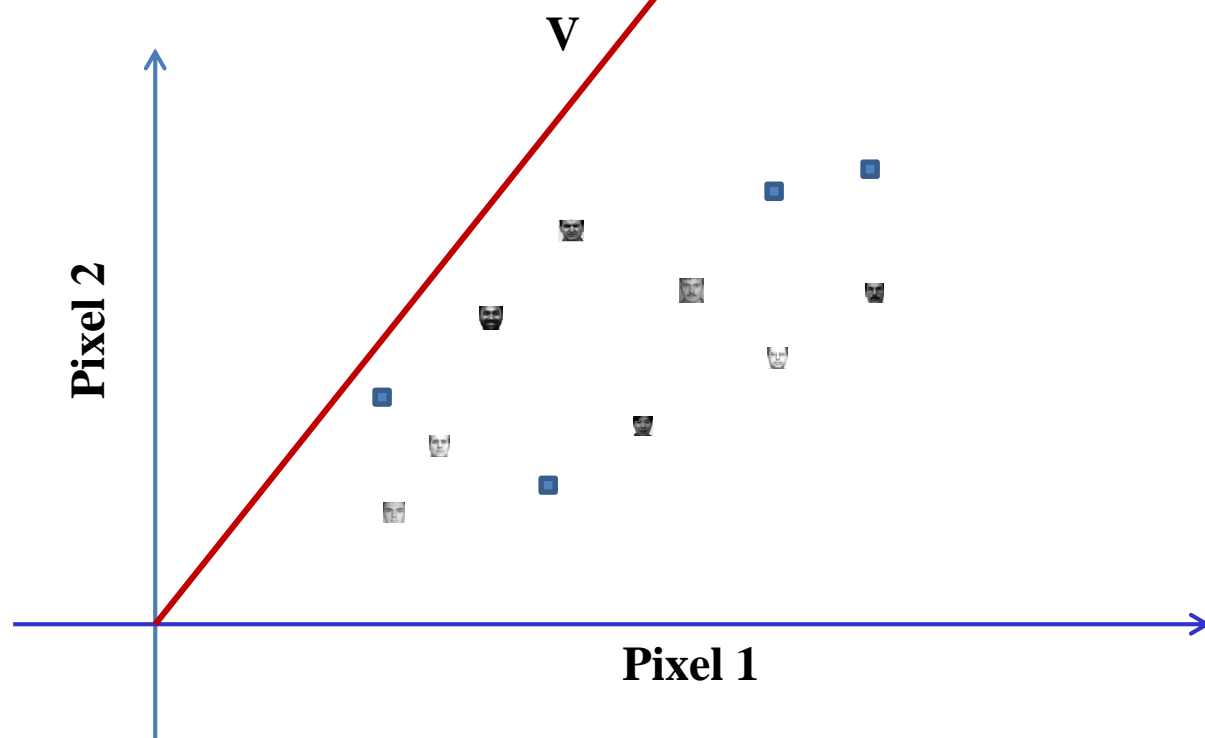
Finding the typical face



- Each “point” represents a face in “pixel space”

Abstracting the problem:

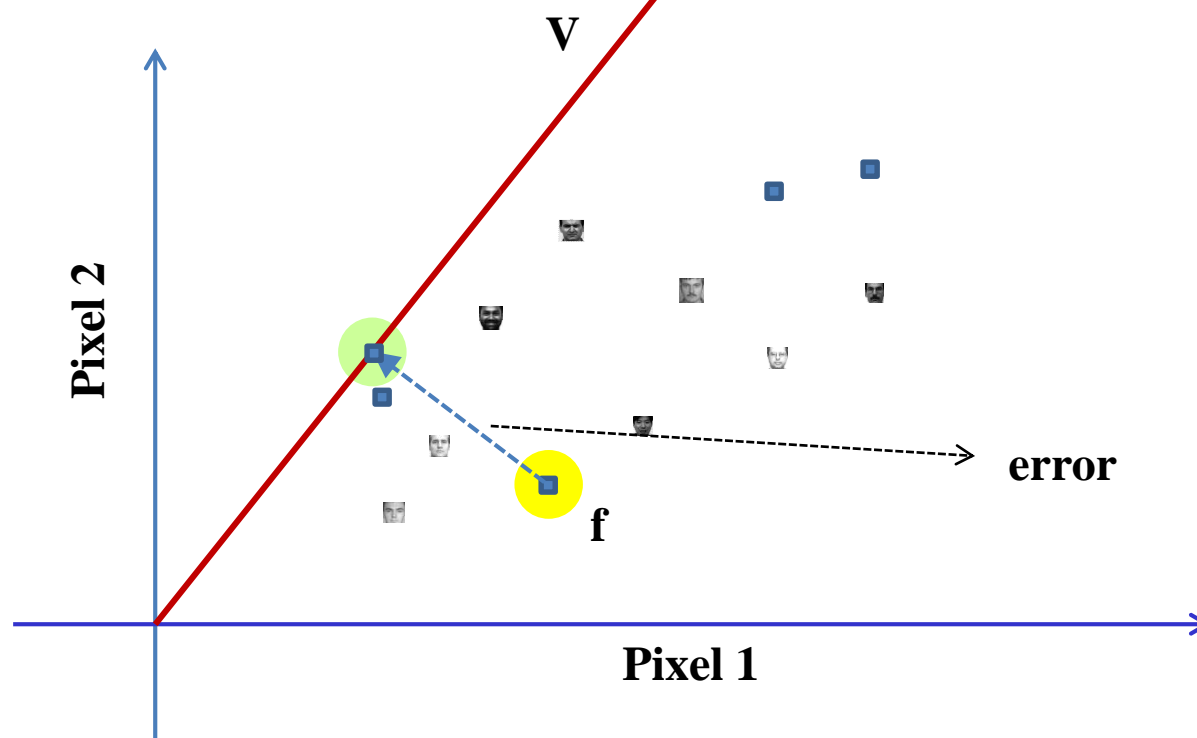
Finding the typical face



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

Abstracting the problem:

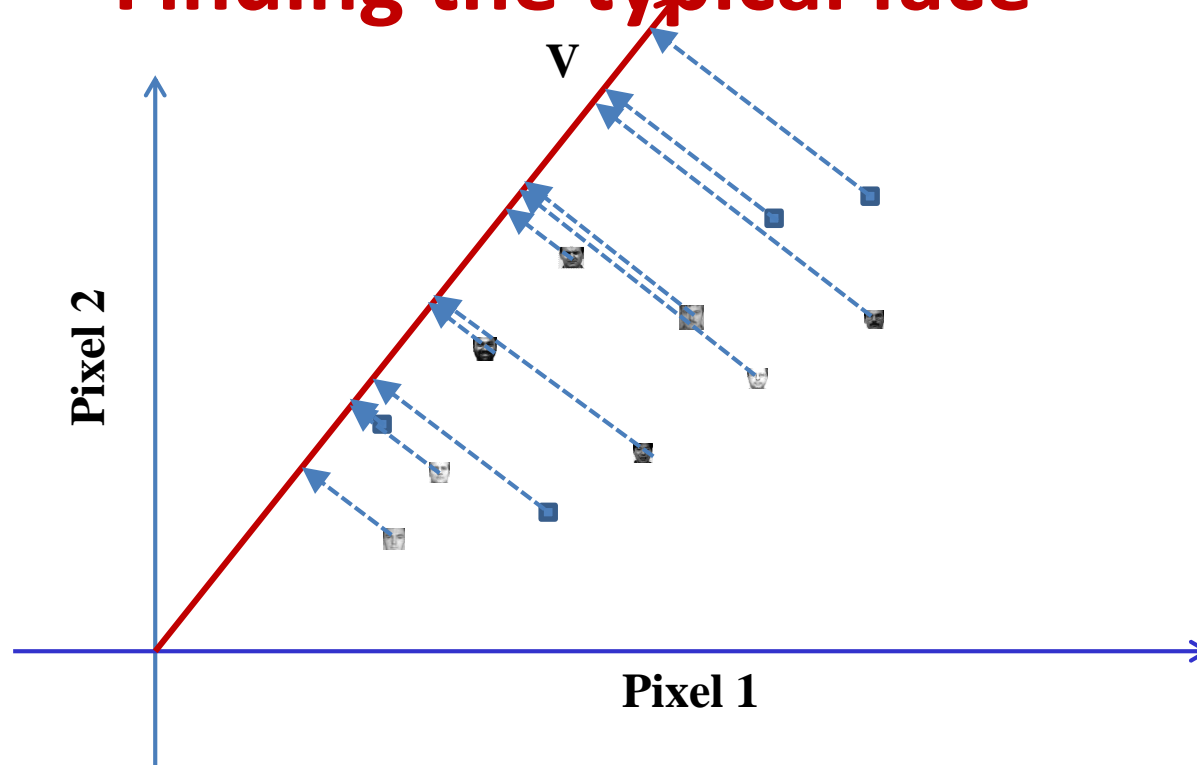
Finding the typical face



- 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:

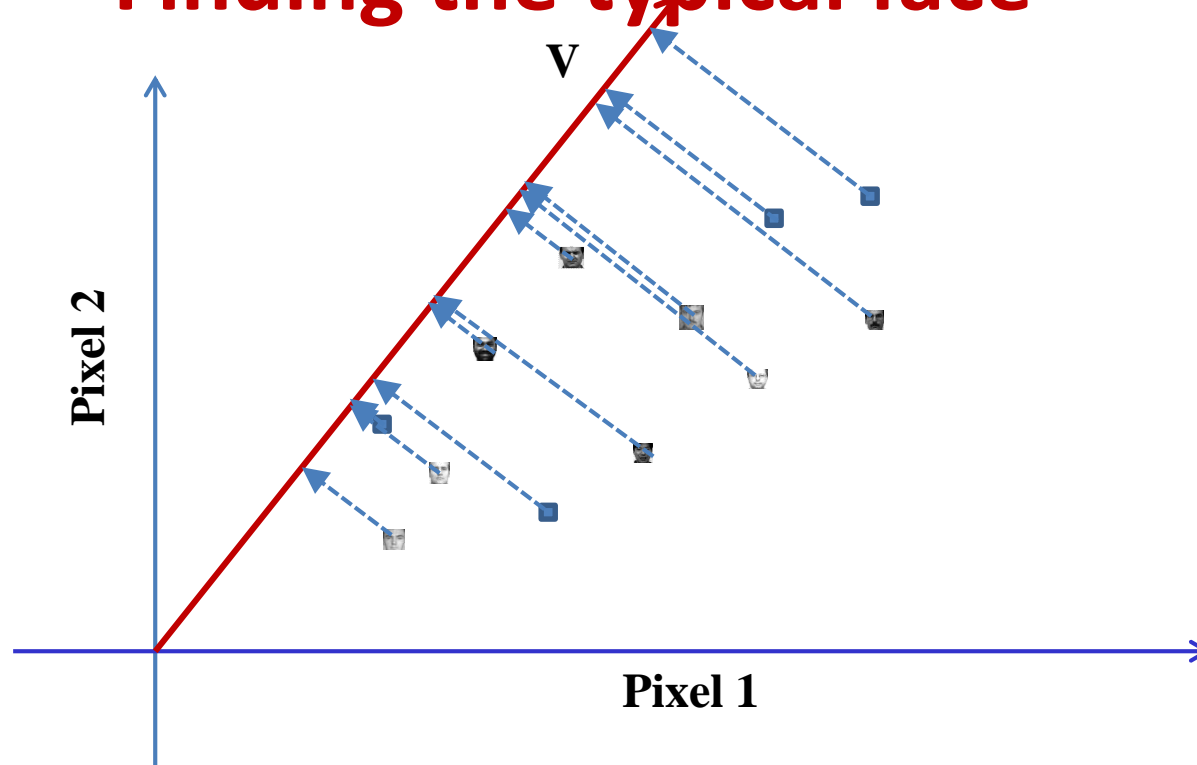
Finding the typical face



- *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:

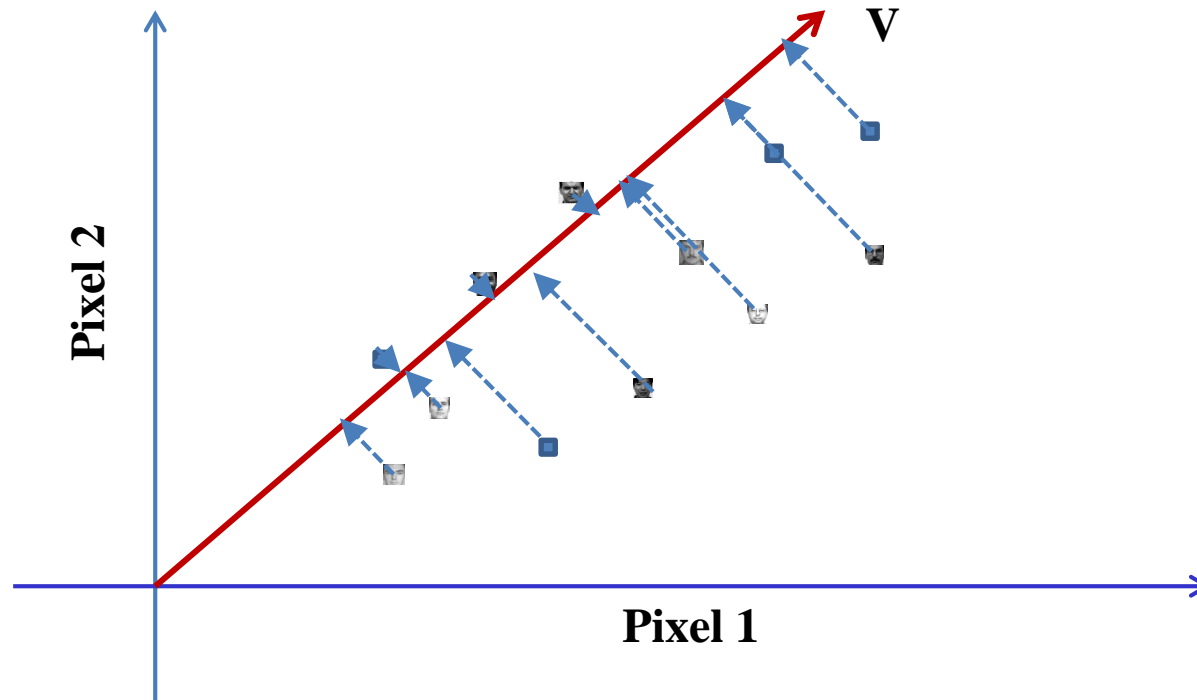
Finding the typical face



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

Abstracting the problem:

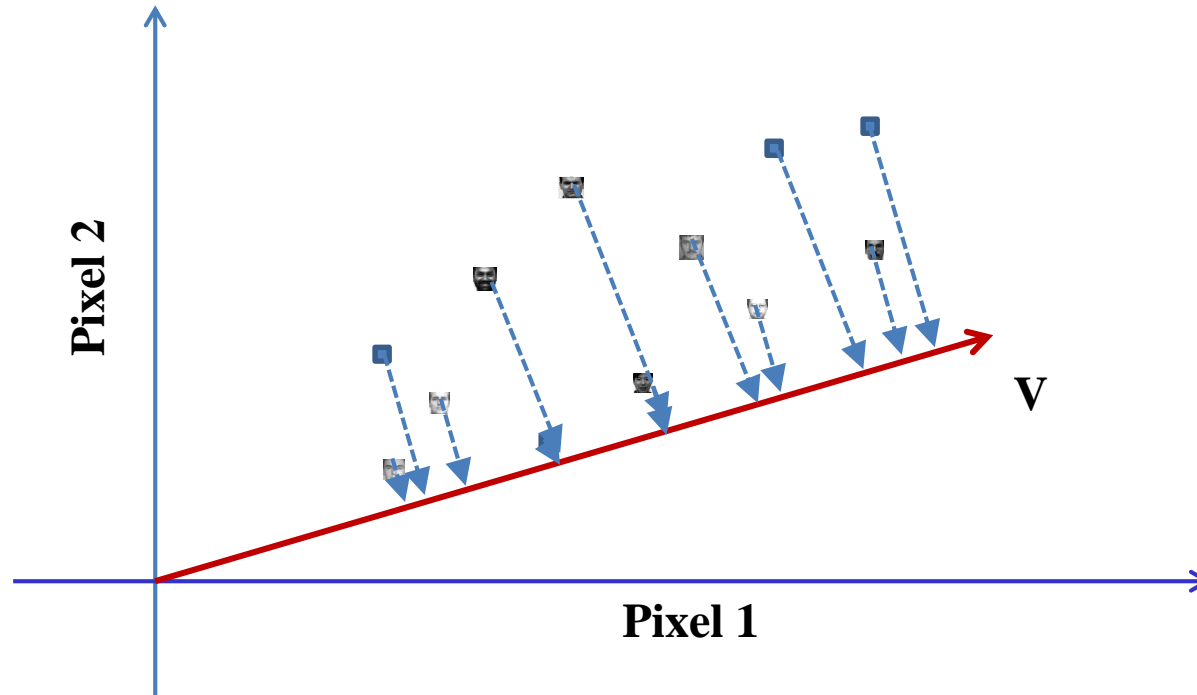
Finding the typical face



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

Abstracting the problem:

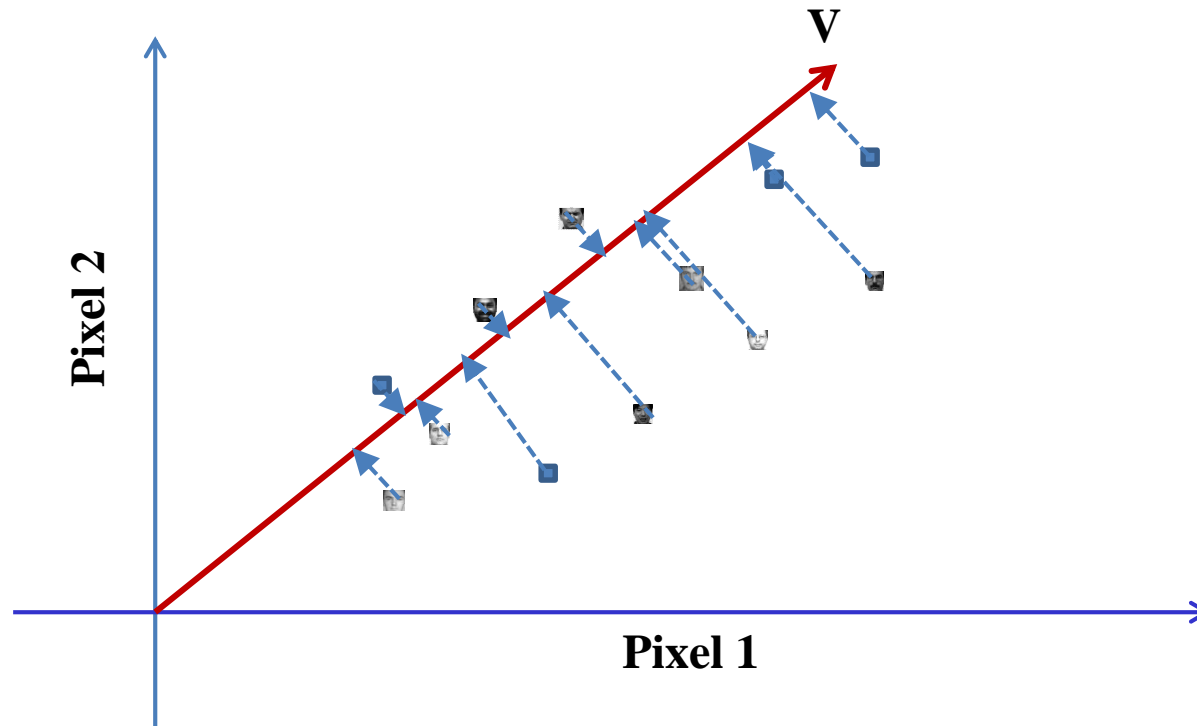
Finding the typical face



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

Abstracting the problem:

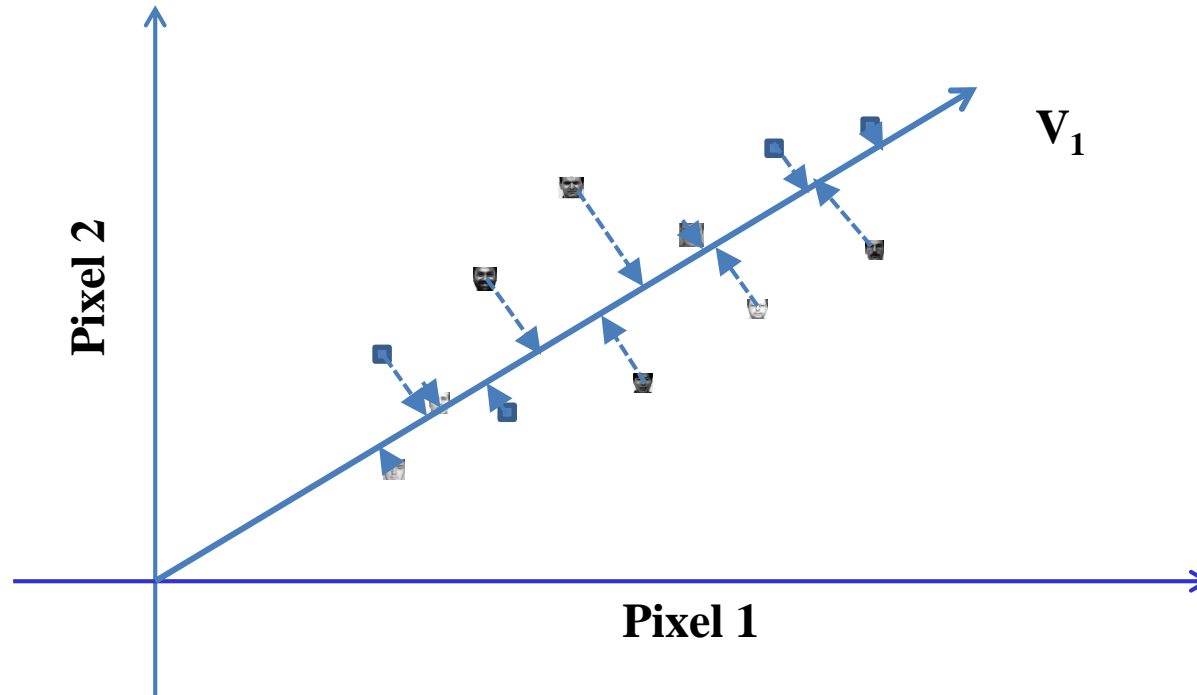
Finding the typical face



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

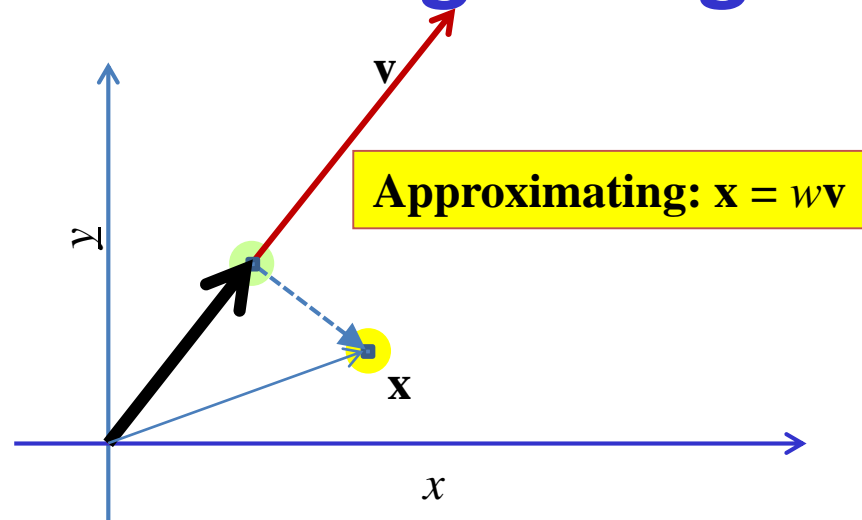
Abstracting the problem:

Finding the typical face



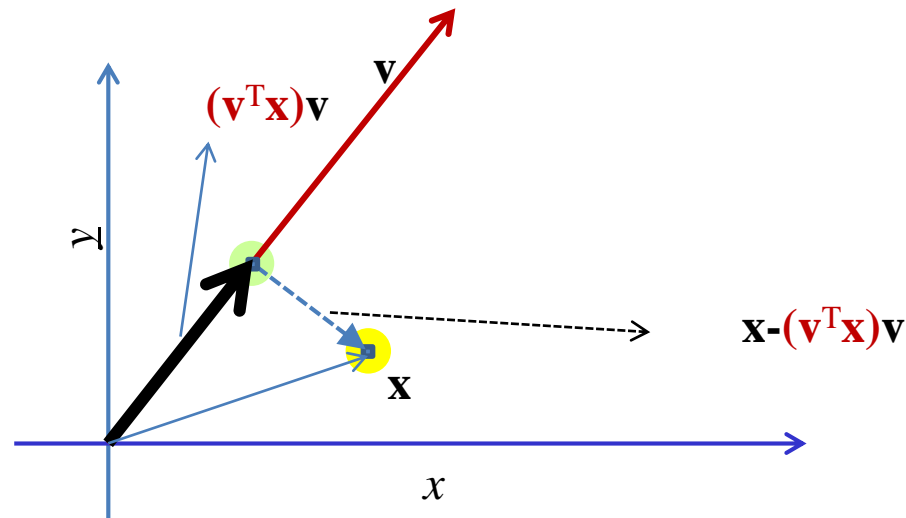
- The problem of finding the first typical face V_1 :
Find the V for which the total projection error is minimum!
- This “minimum squared error” V is our “best” first typical face
- **It is also the first *Eigen face***

Formalizing the Problem: Error from approximating a single vector



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

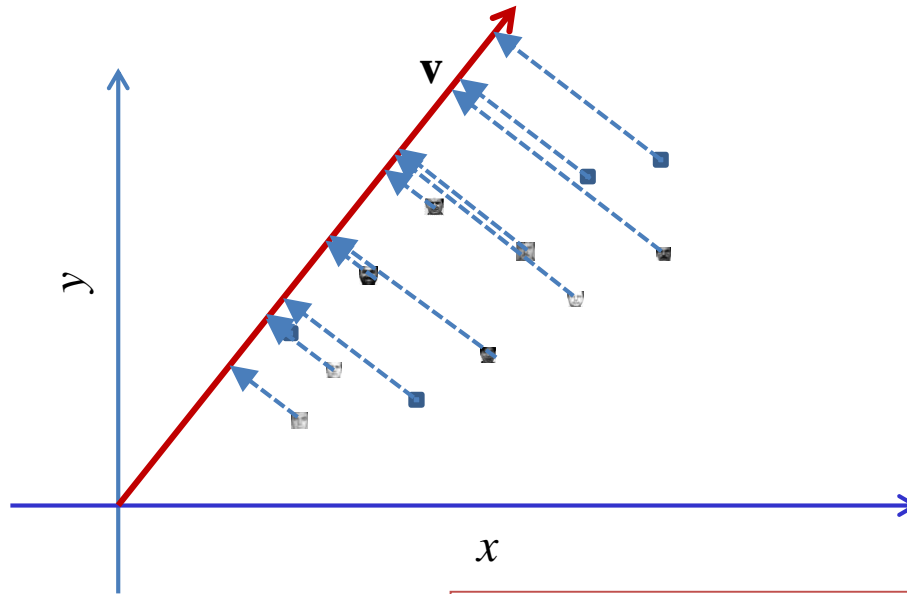
Error from approximating a single vector



- Projection $\hat{\mathbf{x}} = (\mathbf{x}^T \mathbf{v})\mathbf{v}$
- Squared length of projection
$$\|\hat{\mathbf{x}}\|^2 = (\mathbf{x}^T \mathbf{v})^2 = (\mathbf{x}^T \mathbf{v})^T (\mathbf{x}^T \mathbf{v}) = \mathbf{v}^T \mathbf{x} \mathbf{x}^T \mathbf{v}$$
- Pythagoras theorem: Squared length of error $e(\mathbf{x}) = \|\mathbf{x}\|^2 - \|\hat{\mathbf{x}}\|^2$

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

Error for *many* vectors



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

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

- **Goal: Estimate \mathbf{v} to minimize this error!**

Definition: The correlation matrix

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

- The encircled term is the *correlation matrix*

$$\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_N]$$

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

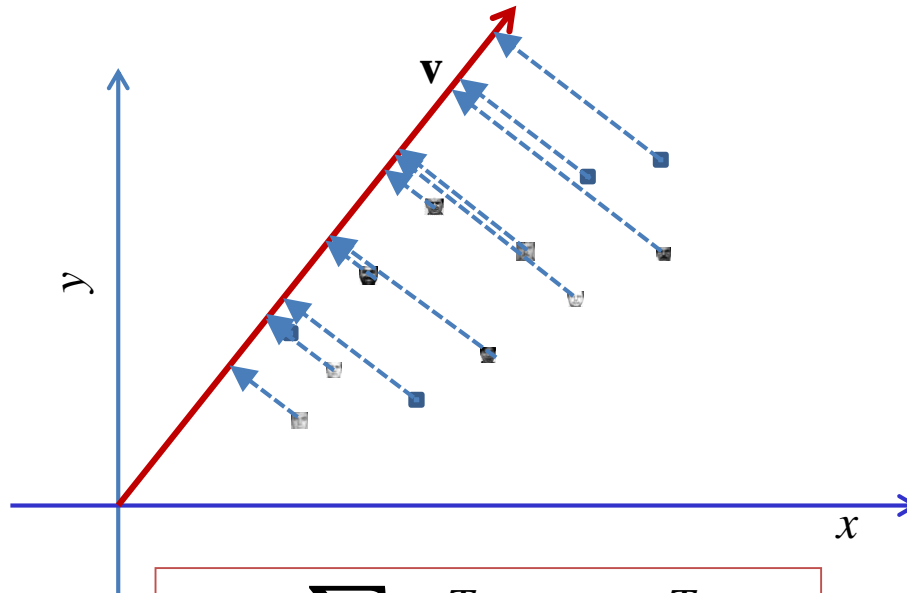
\mathbf{X} = Data Matrix

\mathbf{X}^T = Transposed
Data Matrix

=

Correlation

Error for *many* vectors



- Total error: $E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \mathbf{R} \mathbf{v}$
- Add constraint: $\mathbf{v}^T \mathbf{v} = 1$
- Constrained objective to minimize:

$$L = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \mathbf{R} \mathbf{v} + \lambda (\mathbf{v}^T \mathbf{v} - 1)$$

Two Matrix Identities

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

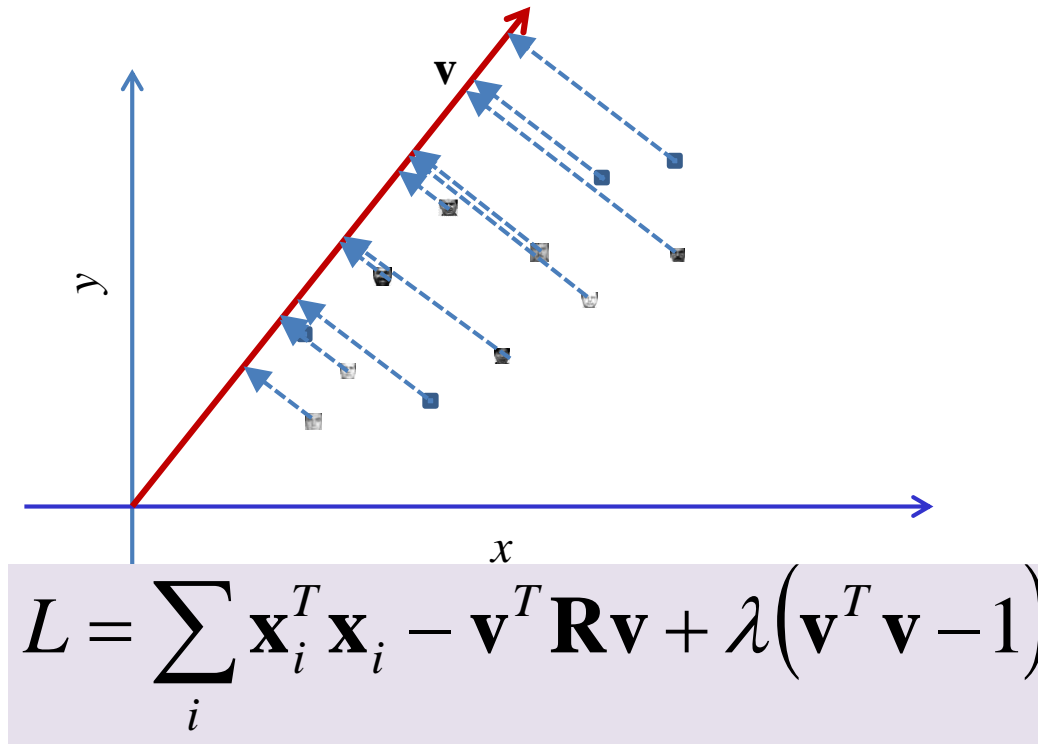
$$L = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \mathbf{v}^T \mathbf{R} \mathbf{v} + \lambda (\mathbf{v}^T \mathbf{v} - 1)$$

$$\nabla_{\mathbf{v}} (\mathbf{v}^T \mathbf{v}) = 2\mathbf{v}$$

$$\nabla_{\mathbf{v}} \mathbf{v}^T \mathbf{R} \mathbf{v} = 2\mathbf{R} \mathbf{v}$$

These are actually *transposed* derivatives, but that does not affect the overall result that follows

Minimizing error

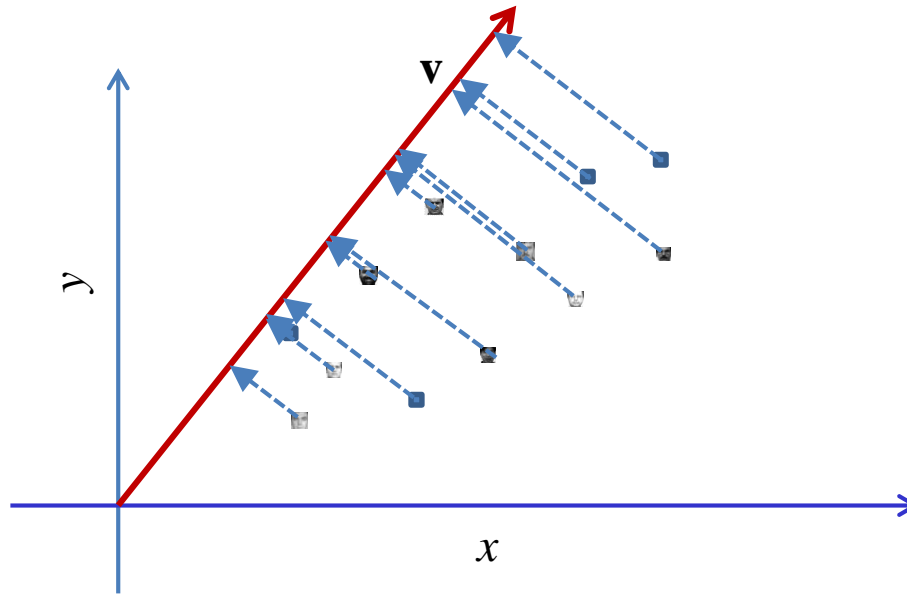


- Differentiating w.r.t \mathbf{v} and equating to 0

$$-2\mathbf{R}\mathbf{v} + 2\lambda\mathbf{v} = 0$$

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

The best “basis”



- The minimum-error basis is found by solving
$$\mathbf{R}\mathbf{v} = \lambda\mathbf{v}$$
- \mathbf{v} is an Eigen vector of the correlation matrix \mathbf{R}
 - λ is the corresponding Eigen value

What about the total error?

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

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

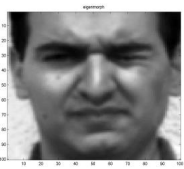
$$E = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \lambda$$

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

Poll 1

The typical face

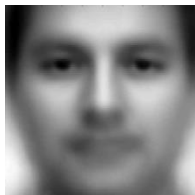
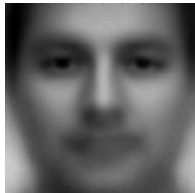
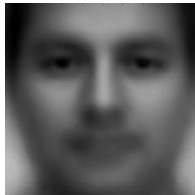
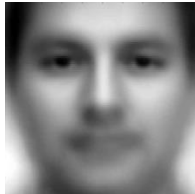


The typical face



- Compute the correlation matrix for your data
 - Arrange them in matrix \mathbf{X} and compute $\mathbf{R} = \mathbf{X}\mathbf{X}^T$
- Compute the *principal* Eigen vector of \mathbf{R}
 - The Eigen vector with the largest Eigen value
 - Explains most of the “energy” in the faces
- 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
- Can we do better?

The *second* typical face



The *first* typical face

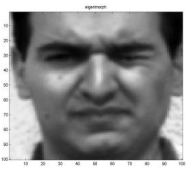


The *second* typical face?



- Approximation with only one typical face V_1 has error
 - Approximating every face as $f = w_{f1} V_1$ is incomplete
- Lets add *second* face to explain this error
 - Add a *second* typical face V_2 . Explain each face now as
 - $f = w_{f1} V_1 + w_{f2} V_2$
- How do we find this second face?

Solution: Iterate



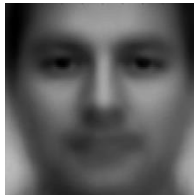
-



=



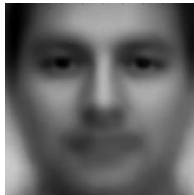
-



=



-



=



-

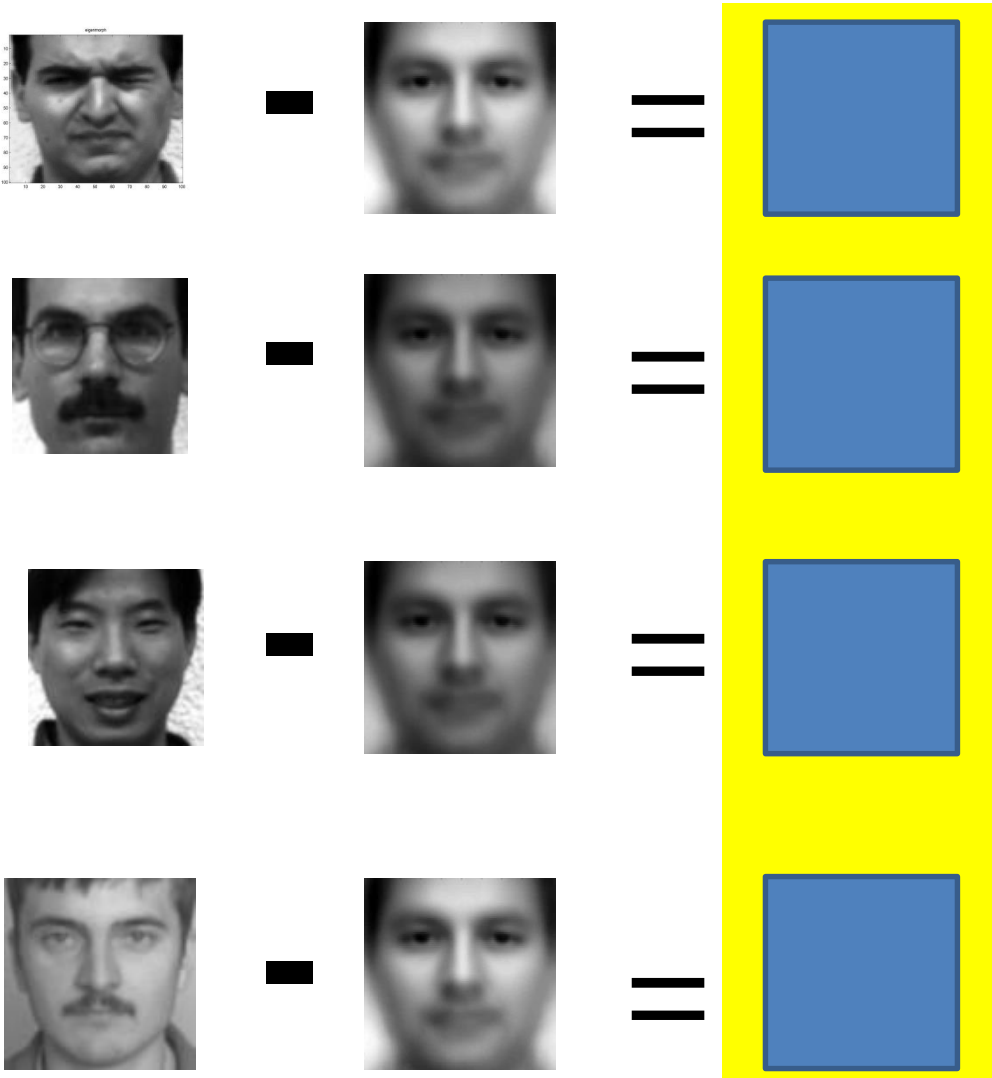


=



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

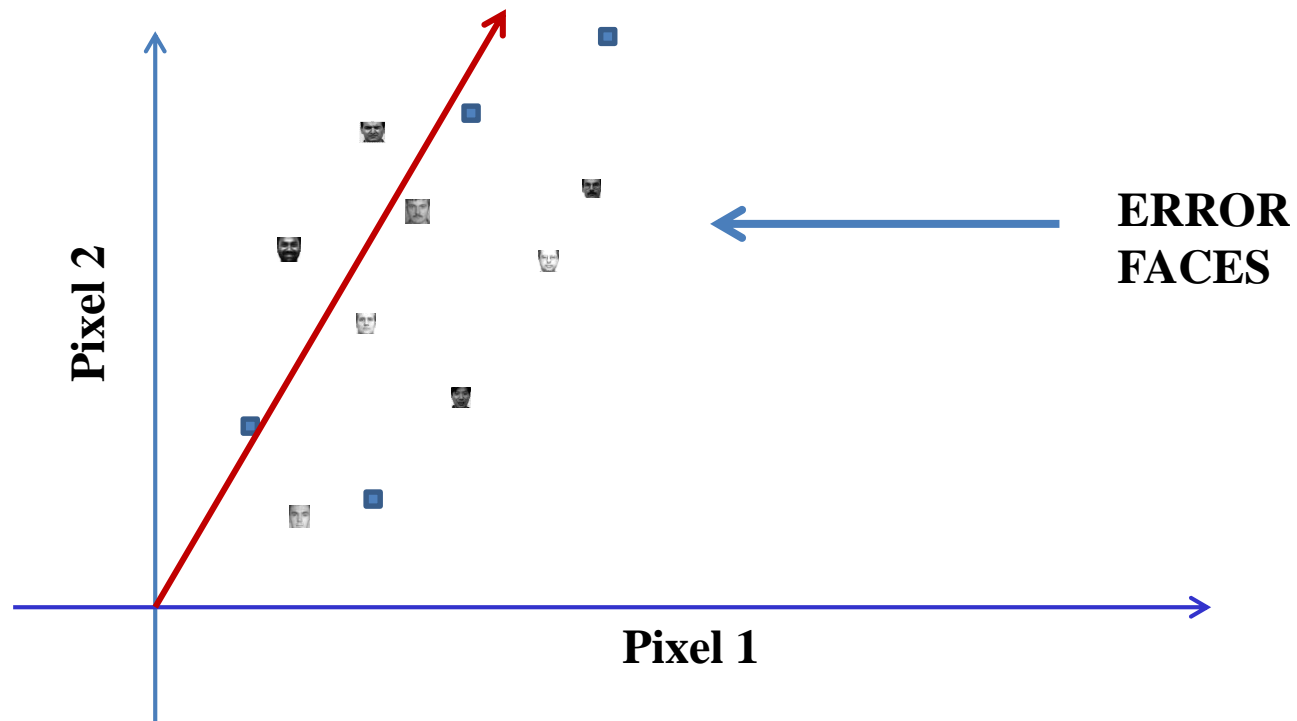
Solution: Iterate



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

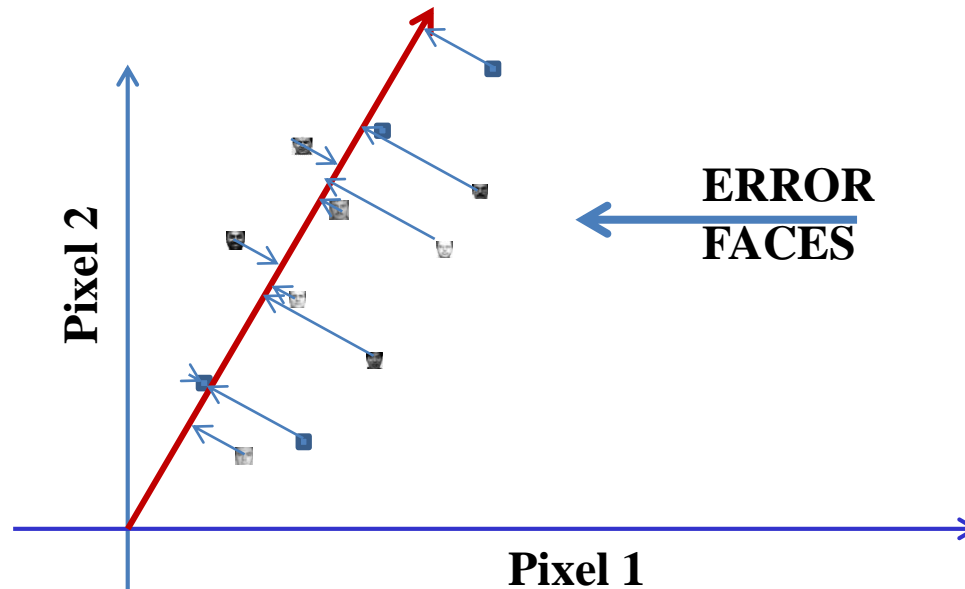
Abstracting the problem:

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*

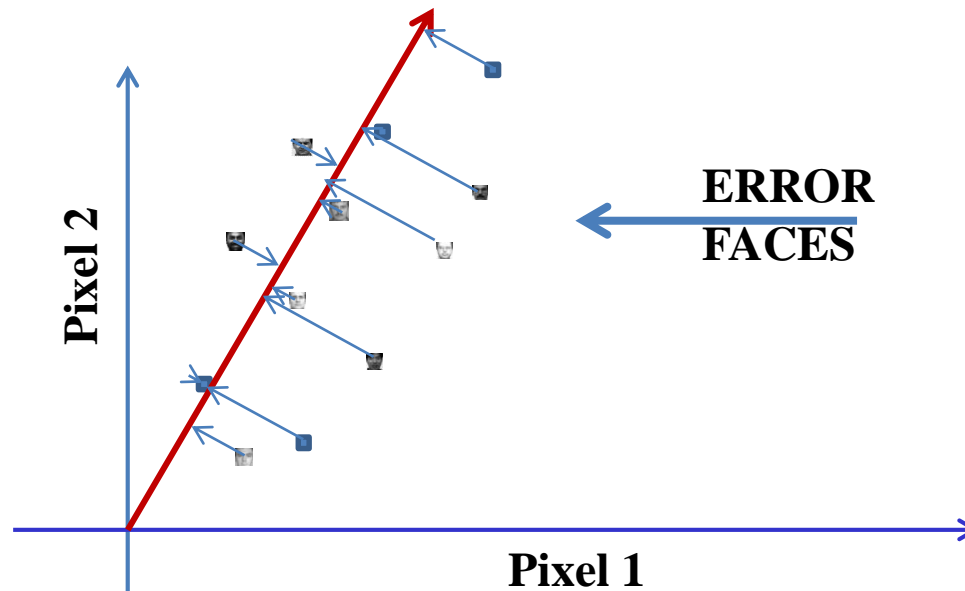
$$L = \sum_i \mathbf{e}_i^T \mathbf{e}_i - \mathbf{v}^T \sum_i \mathbf{e}_i \mathbf{e}_i^T \mathbf{v} + \lambda (\mathbf{v}^T \mathbf{v} - 1)$$

- Defining the autocorrelation of the error

$$\mathbf{R}_e = \sum \mathbf{e} \mathbf{e}^T$$

$$L = \sum_i \mathbf{e}_i^T \mathbf{e}_i - \mathbf{v}^T \mathbf{R}_e \mathbf{v} + \lambda (\mathbf{v}^T \mathbf{v} - 1)$$

Minimizing error



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

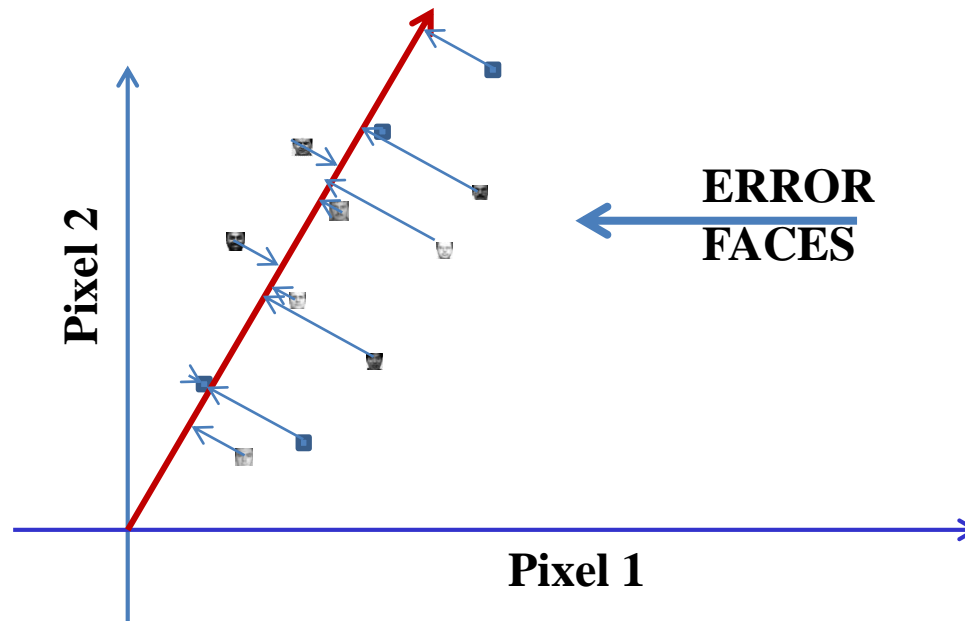
$$L = \sum_i \mathbf{e}_i^T \mathbf{e}_i - \mathbf{v}^T \mathbf{R}_e \mathbf{v} + \lambda (\mathbf{v}^T \mathbf{v} - 1)$$

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

$$-2\mathbf{R}_e \mathbf{v} + 2\lambda \mathbf{v} = 0$$

$$\mathbf{R}_e \mathbf{v} = \lambda \mathbf{v}$$

Minimizing error



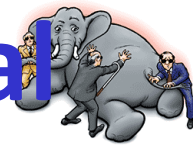
The same math applies
but now to the set
of *error data points*

- The minimum-error basis is found by solving

$$\mathbf{R}_e \mathbf{v}_2 = \lambda \mathbf{v}_2$$

- \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

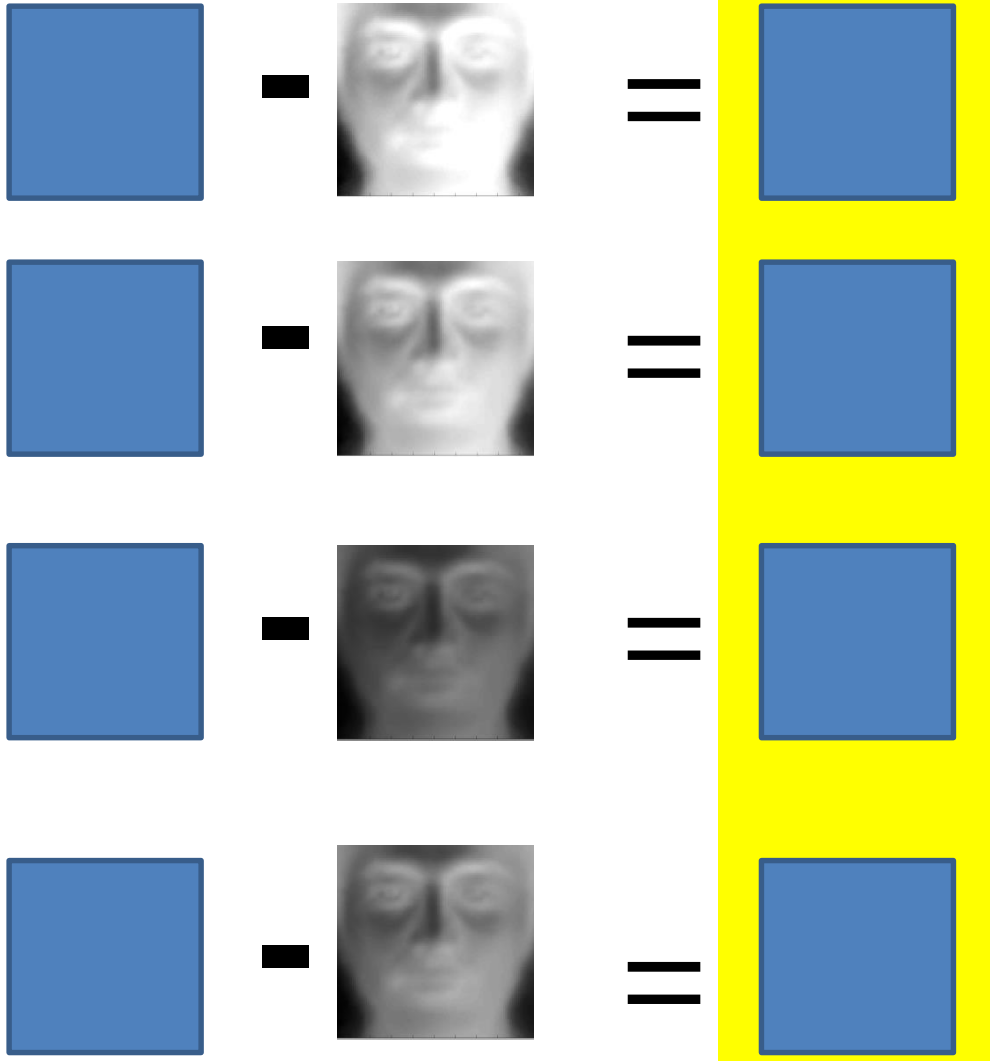


- But approximation with the two faces will *still* result in error
- So we need more typical faces to explain *this* error
- We can do this by subtracting the appropriately scaled version of the second “typical” face from the error images and repeating the process

Solution: Iterate

Error face

Second-level error



- 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

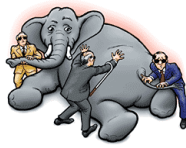
An interesting property

- 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
- So what are we really doing?

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 $f = w_{f,1} V_1 + w_{f,2} V_2 + w_{f,3} V_3 + \dots + w_{f,k} V_k$
 - V_2 is used to “correct” errors resulting from using only V_1 . So on average

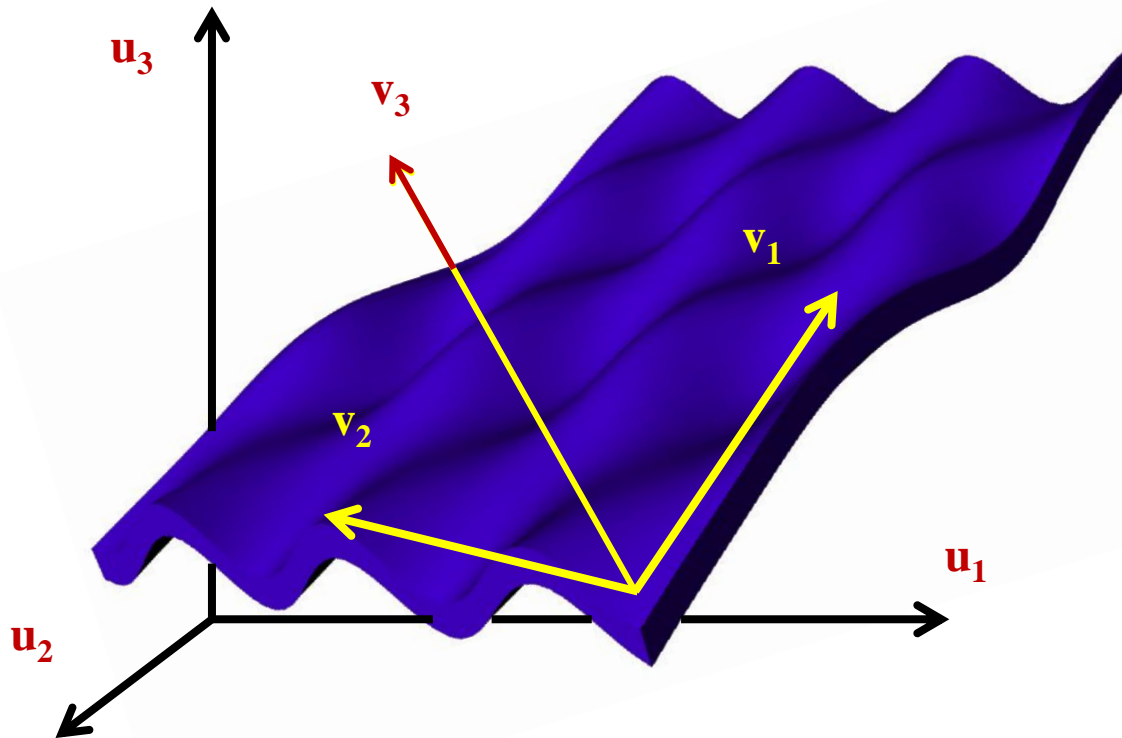
$$\|f - (w_{f,1} V_{f,1} + w_{f,2} V_{f,2})\|^2 < \|f - w_{f,1} V_{f,1}\|^2$$

- V_3 corrects errors remaining after correction with V_2

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

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

Recall: Basis based representation



- *The most important challenge in ML: Find the best set of bases for a given data set*

The Energy Compaction Property

- Define “best”?
- The description

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

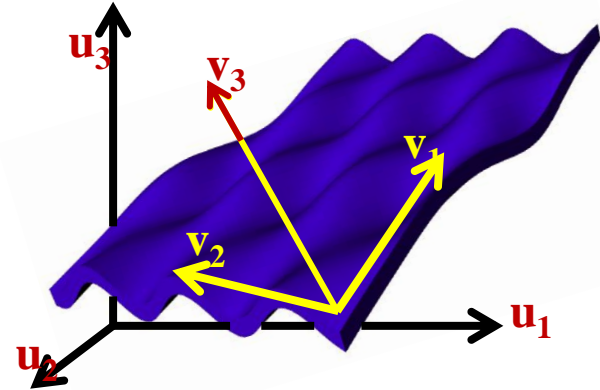
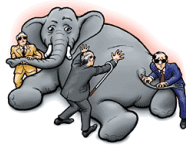
- The ideal:

$$\hat{X}_i \approx w_1 B_1 + w_2 B_2 + \dots + w_i B_i \quad \text{Error}_i = \|X - \hat{X}_i\|^2$$

$$\text{Error}_i < \text{Error}_{i-1}$$

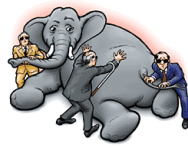
- If the description is terminated at any point, we should still get most of the information about the data
 - No other set of bases (for any leading subset of bases) should result in lower Error for the same number of bases

Finding the bases

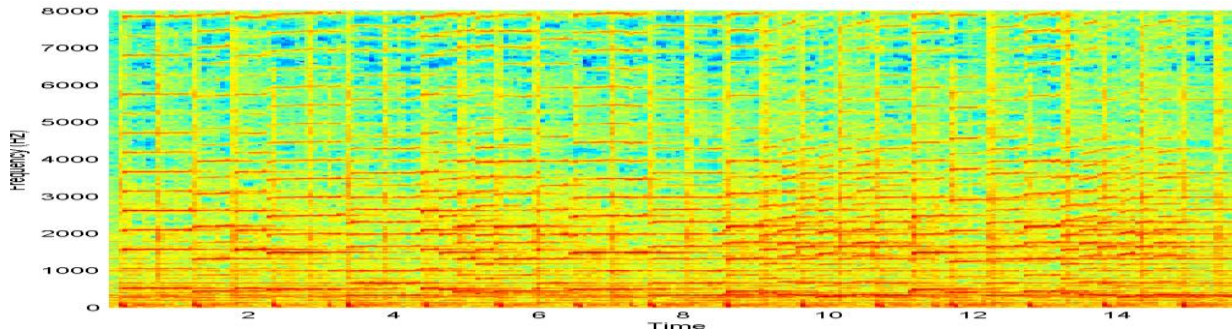


- Finding the optimal set of “typical faces” in this example is the problem of finding the optimal basis set for the data

A recollection

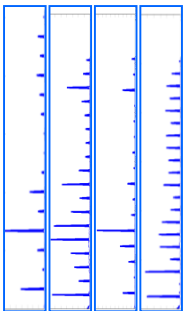


M =



$$S = \text{Pinv}(N) M$$

N =



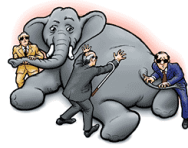
$$U = ?$$

$$U = NS \approx M$$

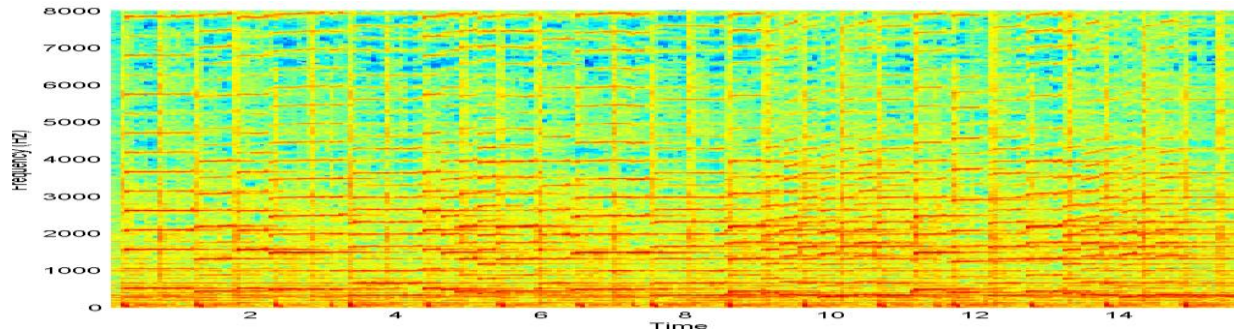
$$S = \text{pinv}(N)M$$

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

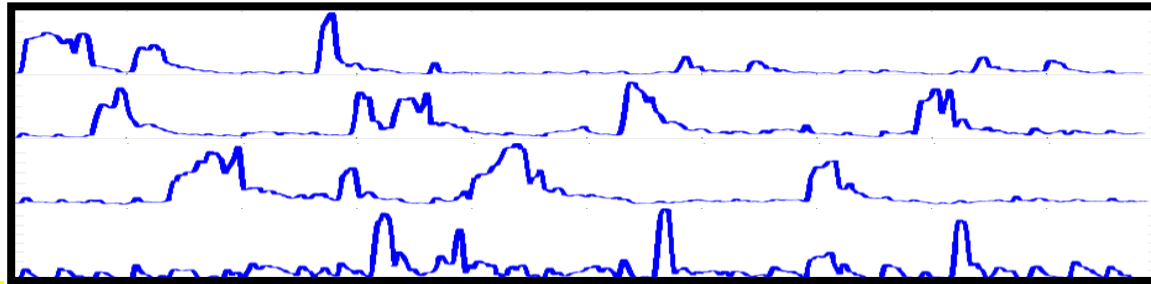
How about the other way?



M =



S =



$$N = M \text{Pinv}(S)$$

N =

?

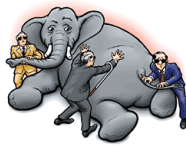
U = ?

$$U = NS \approx M$$

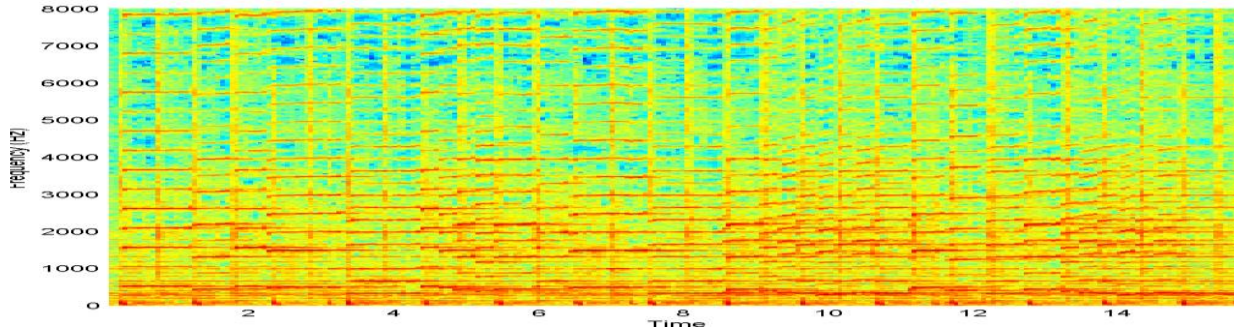
$$N = M \text{pinv}(S)$$

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

Finding Everything



$M =$



$S =$

?

$N =$

?

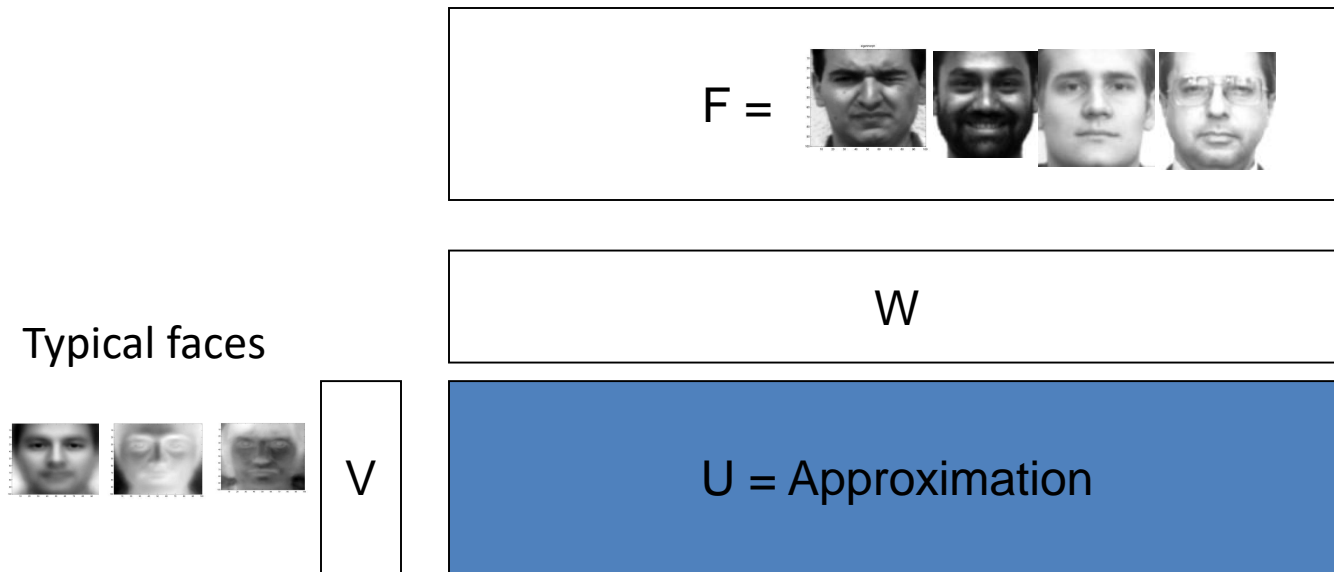
$U =$

?

$$U = NS \approx M$$

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

The Same Problem



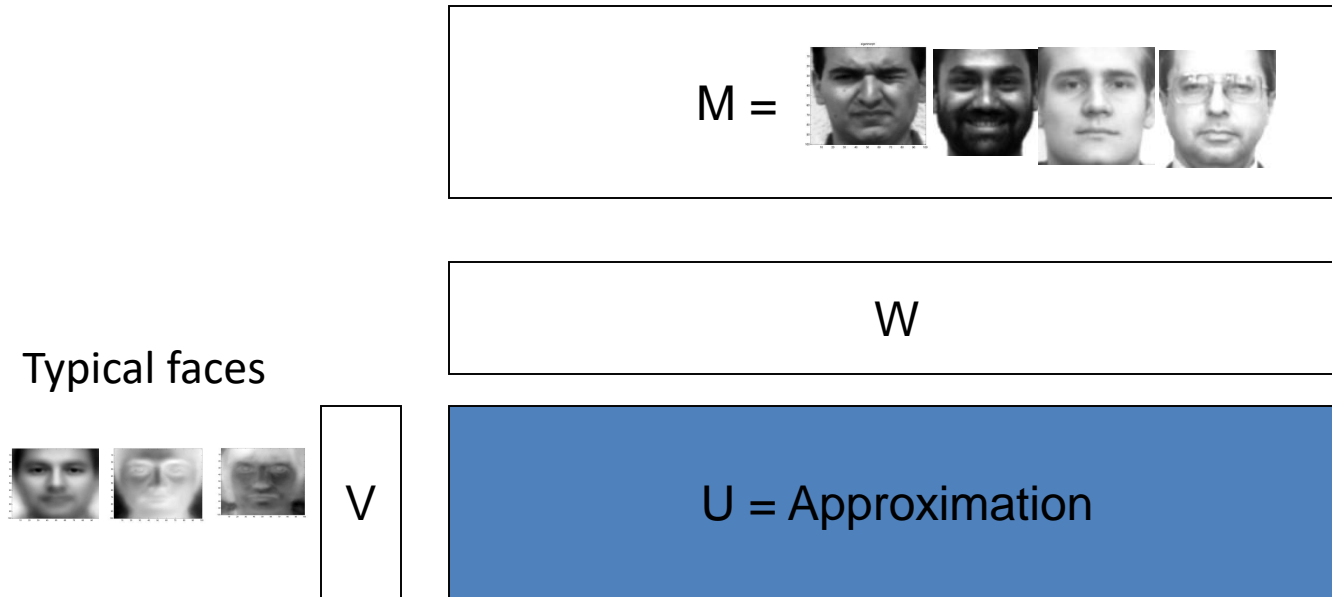
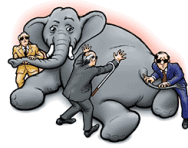
- Here U , V and W are all unknown and must be estimated
 - Such that the total squared error between F and U is minimized
- For each face f
 - $f = w_{f,1}V_1 + w_{f,2}V_2 + \dots + w_{f,K}V_K$
- For the collection of faces $F \approx VW$
 - V is $D \times K$, W is $K \times N$
 - D is the number of pixels, N is the number of faces in the set

Poll 2

Finding the bases

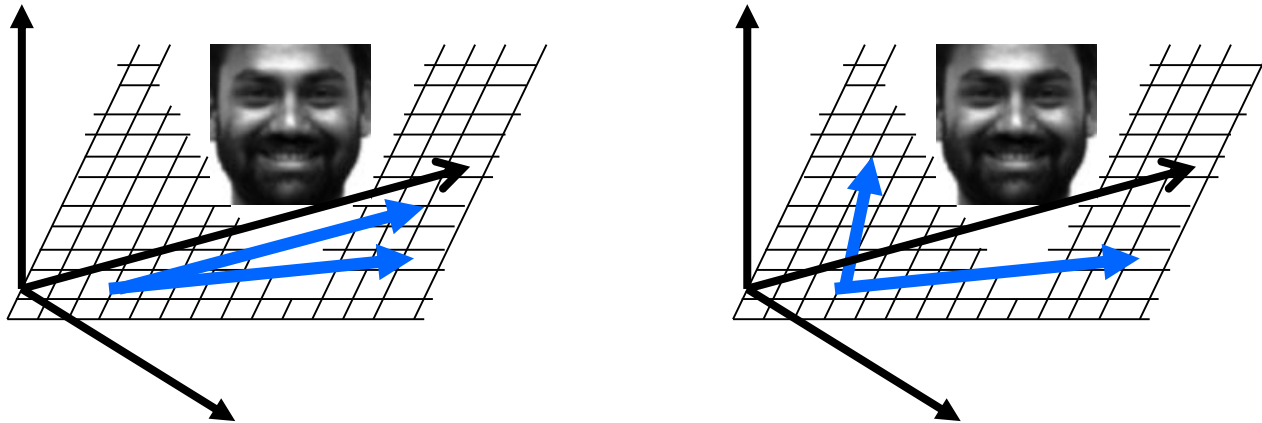
- We just saw an incremental procedure for finding the bases
 - Finding one new basis at a time that explains residual error not explained by previous bases
 - An instance of a procedure is called “Gram-Schmidt” orthogonalization
- We can also do it all at once

With many typical faces



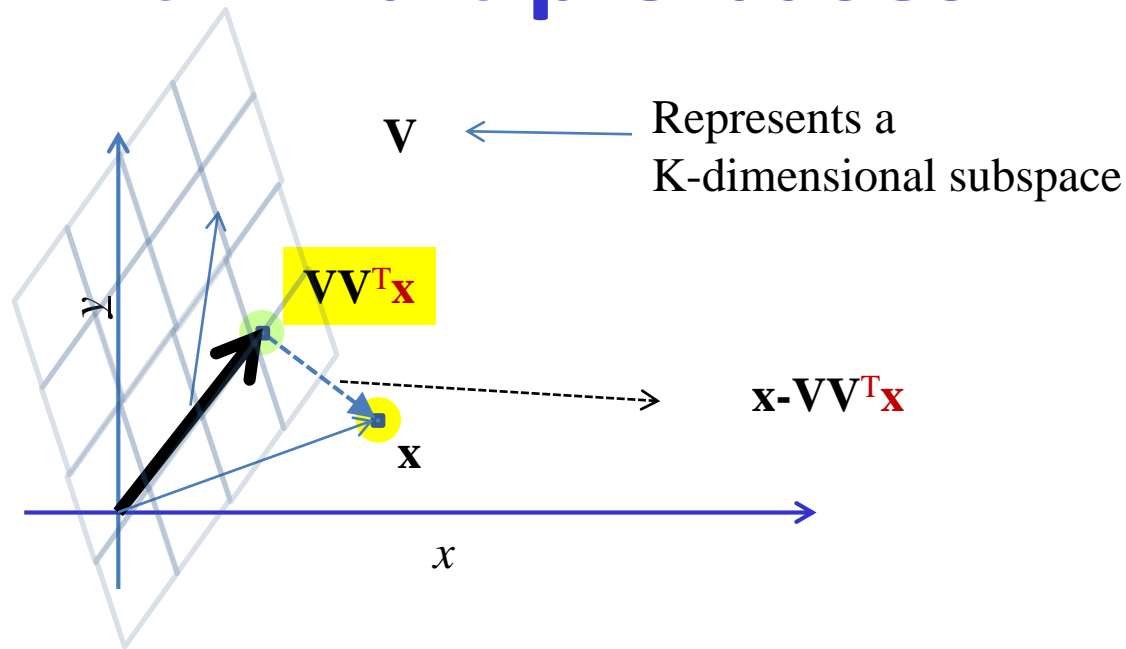
- Approximate **every** face f as $f = w_{f,1} V_1 + w_{f,2} V_2 + \dots + w_{f,k} V_k$
- Here W , V and U are ALL unknown and must be determined
 - Such that the squared error between U and M is minimum

With multiple bases



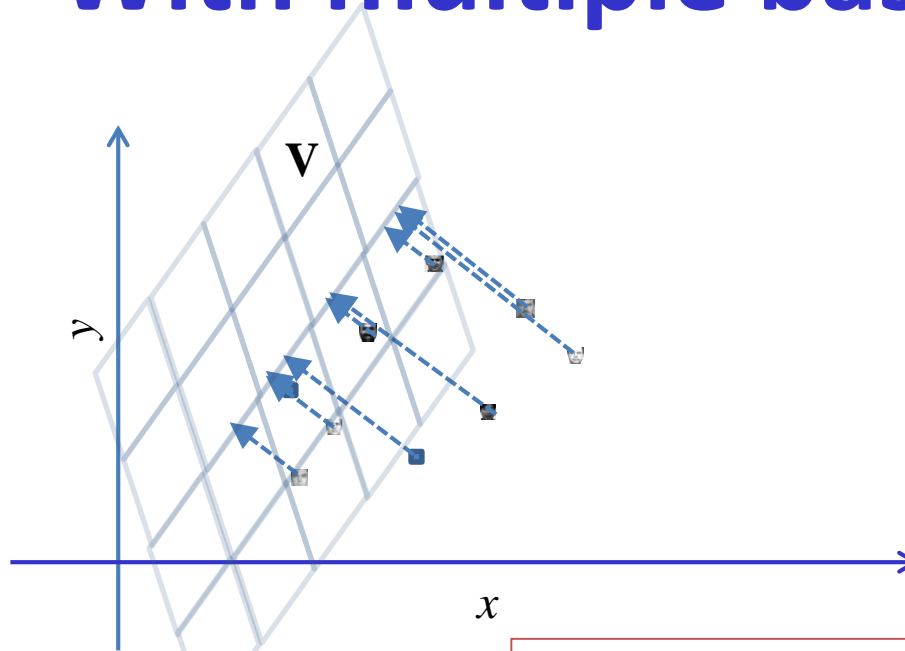
- **Assumption: all bases $\mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3 \dots$ are unit length**
- **Assumption: all bases are orthogonal to one another: $\mathbf{v}_i^T \mathbf{v}_j = 0$ if $i \neq j$**
 - We are trying to find the optimal K-dimensional subspace to project the data
 - Any set of basis vectors in this subspace will define the subspace
 - Constraining them to be orthogonal does not change this
- I.e. if $\mathbf{V} = [\mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3 \dots]$, $\mathbf{V}^T \mathbf{V} = \mathbf{I}$
 - $\text{Pinv}(\mathbf{V}) = \mathbf{V}^T$
- 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 \mathbf{V} to minimize this error!**

Minimizing Error

- With constraint $\mathbf{V}^T \mathbf{V} = \mathbf{I}$, we get the modified objective

$$L = \sum_i \mathbf{x}_i^T \mathbf{x}_i - \sum_i \mathbf{x}_i^T \mathbf{V} \mathbf{V}^T \mathbf{x}_i + \text{trace}(\Lambda(\mathbf{V}^T \mathbf{V} - \mathbf{I}))$$

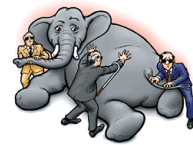
- Λ is a diagonal Lagrangian matrix
- Constraints are $\mathbf{v}_i^T \mathbf{v}_i = 1$ and $\mathbf{v}_i^T \mathbf{v}_j = 0$ for $i \neq j$
- Differentiating w.r.t \mathbf{V} and equating to 0

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

Finding the optimal K bases

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

- Compute the Eigendecomposition 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!

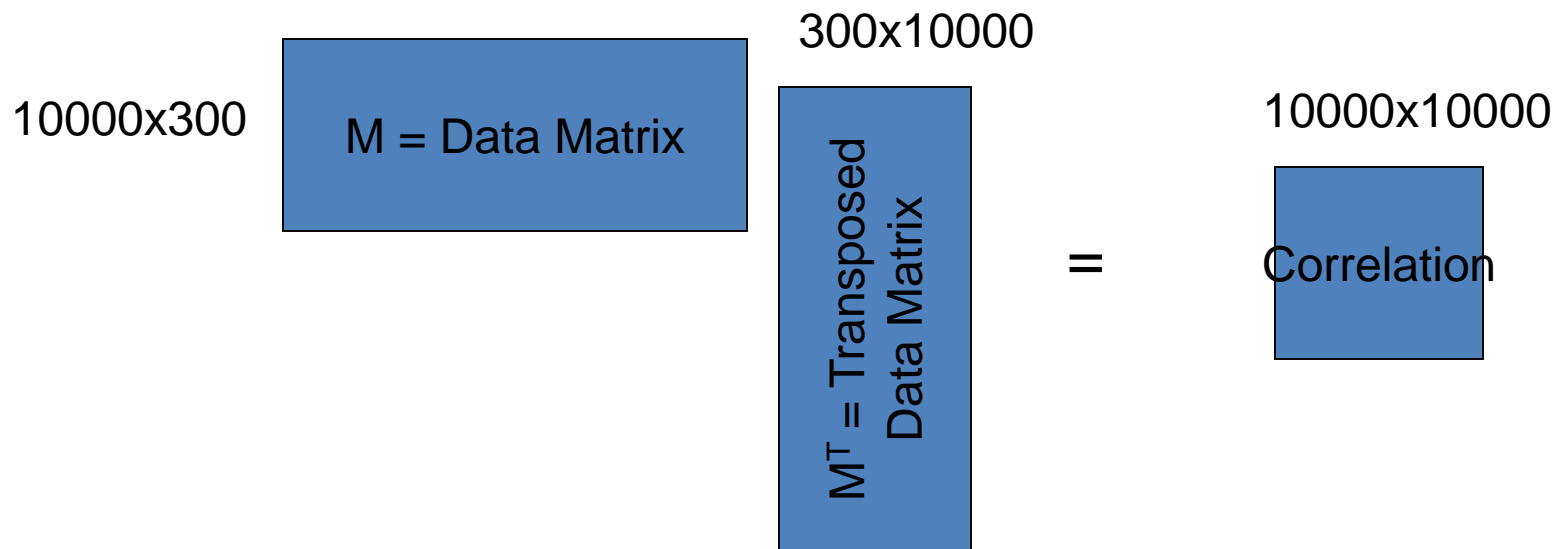


- Arrange your input data into a matrix \mathbf{X}
- Compute the correlation $\mathbf{R} = \mathbf{X}\mathbf{X}^T$
- Solve the Eigen decomposition: $\mathbf{R}\mathbf{V} = \mathbf{\Lambda}\mathbf{V}$
- The Eigen vectors corresponding to the K largest eigen values are our optimal bases
- We will refer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

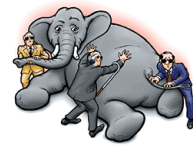
Poll 3



How many Eigen faces



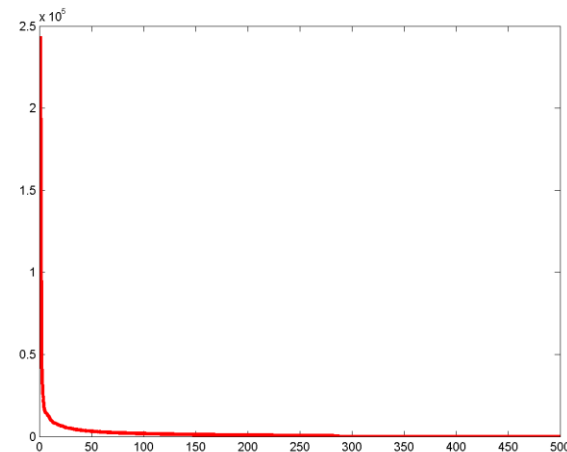
- How to choose “K” (number of Eigen faces)
- Lay all faces side by side in vector form to form a matrix
 - In my example: 300 faces. So the matrix is 10000×300
- Multiply the matrix by its transpose
 - The correlation matrix is 10000×10000



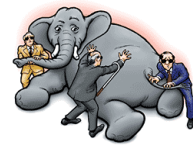
Eigen faces

$[U, S] = \text{eig}(\text{correlation})$

$$S = \begin{bmatrix} \lambda_1 & . & 0 & . & 0 \\ 0 & \lambda_2 & 0 & . & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & 0 & . & \lambda_{10000} \end{bmatrix} \quad U = \begin{bmatrix} \text{eigenface1} \\ \text{eigenface2} \\ \bullet \bullet \bullet \end{bmatrix}$$

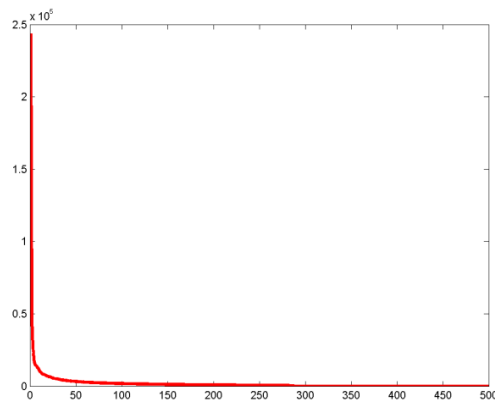


- Compute the eigen vectors
 - Only 300 of the 10000 eigen values are non-zero
 - Why?
- Retain eigen vectors with high eigen values (>0)
 - Could use a higher threshold



Eigen Faces

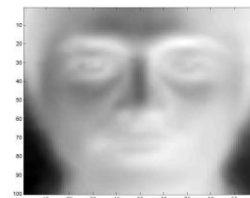
$$U = \begin{bmatrix} \text{eigenface1} \\ \text{eigenface2} \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$



eigenface1

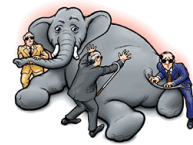


eigenface2



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

A diagram illustrating the representation of a face as a linear combination of eigenfaces. On the left is a grayscale image of a man's face. To its right is an equals sign, followed by a weight w_1 multiplied by a first eigenface (a blurry face), plus a weight w_2 multiplied by a second eigenface (a blurry face with horizontal features), plus a weight w_3 multiplied by a third eigenface (a blurry face with vertical features), followed by three dots indicating further terms in the series.

Representation

$$\begin{pmatrix} \text{face image} \end{pmatrix} = [w_1 \ w_2 \ w_3 \ \dots]^T$$

- 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

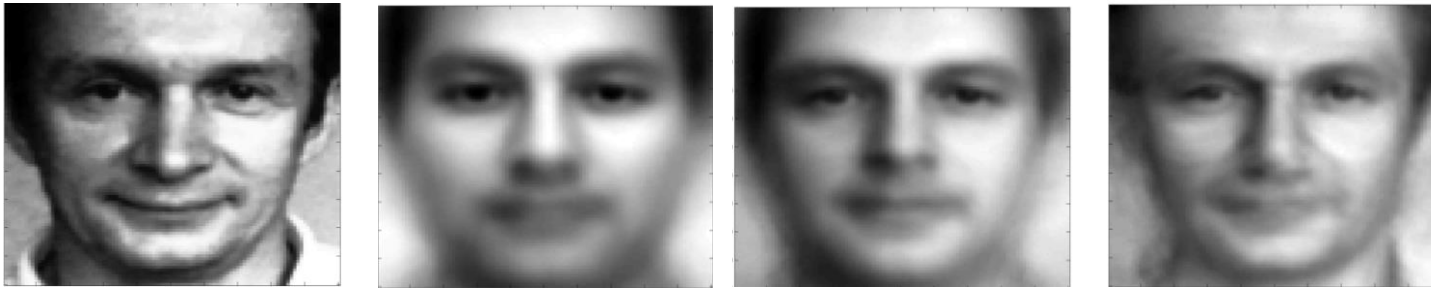


- Approximating a face with 10 eigenfaces:

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

Energy Compaction Example

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

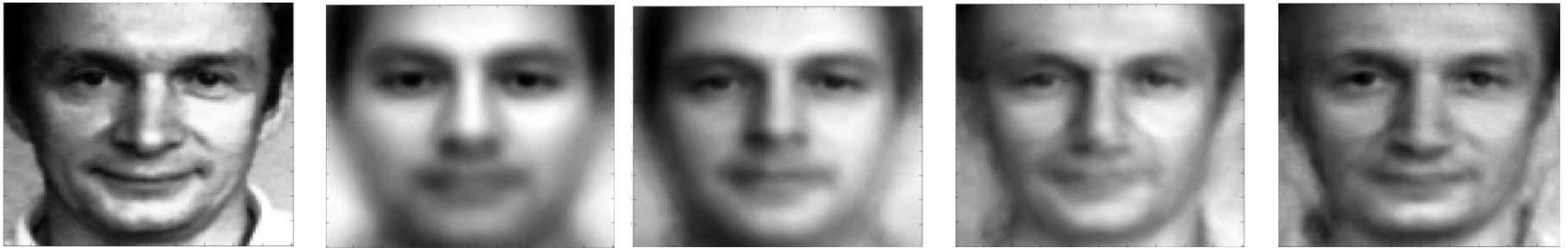


- Approximating a face with 30 eigenfaces:

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

Energy Compaction Example

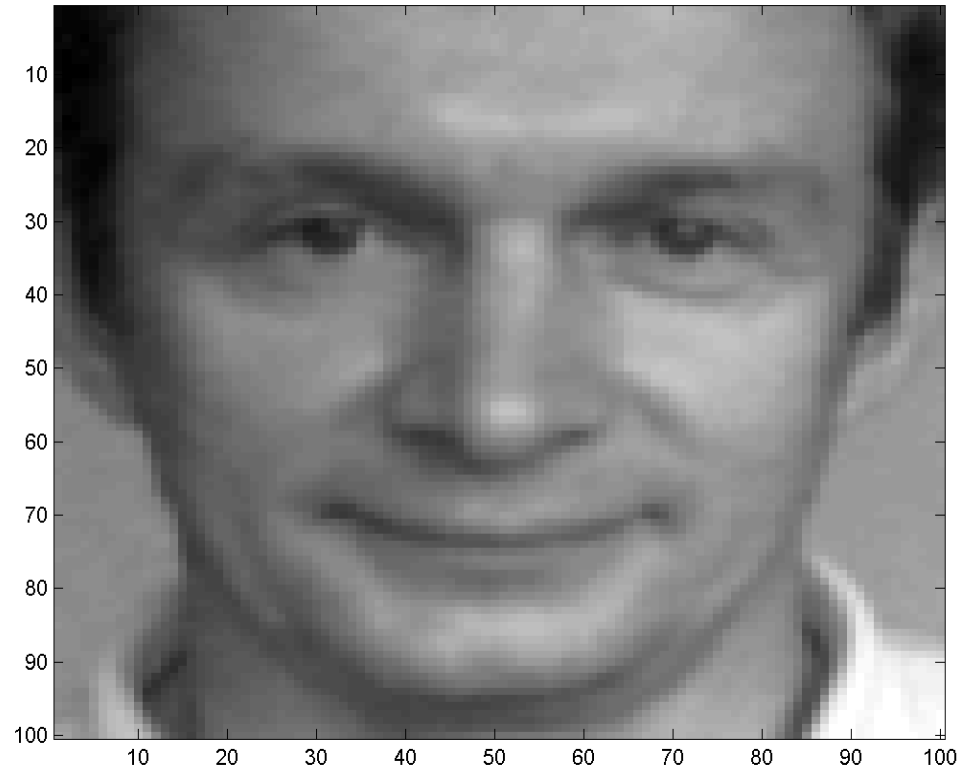
- One outcome of the “energy compaction principle”: the approximations are recognizable



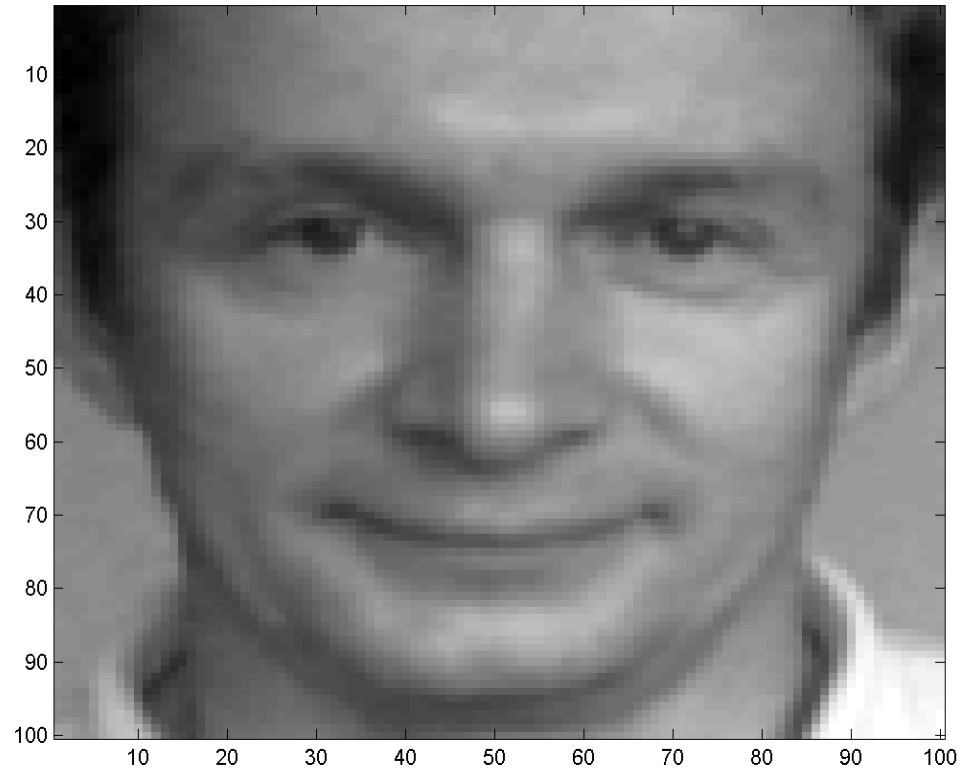
- Approximating a face with 60 eigenfaces:

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

How did I do this?



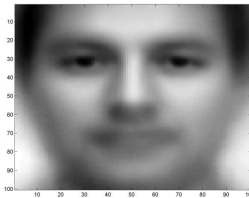
How did I do this?



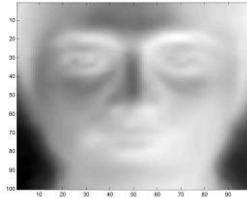
- Hint: only changing weights assigned to Eigen faces..

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

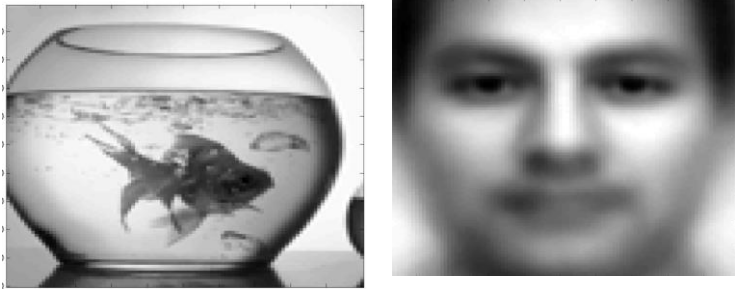
- Eigen bases are class specific



- Composing a fishbowl from Eigenfaces

Class specificity

- Eigen bases are class specific

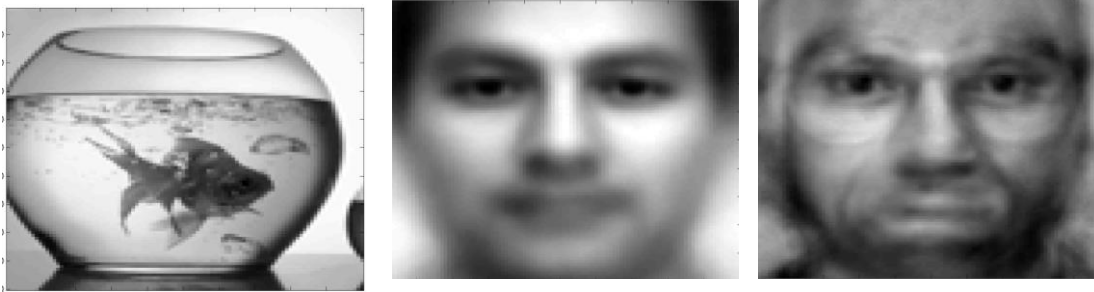


- Composing a fishbowl from Eigenfaces
- With 1 basis

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

Class specificity

- Eigen bases are class specific



- Composing a fishbowl from Eigenfaces
- With 10 bases

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

Class specificity

- Eigen bases are class specific



- Composing a fishbowl from Eigenfaces
- With 30 bases

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

Class specificity

- Eigen bases are class specific

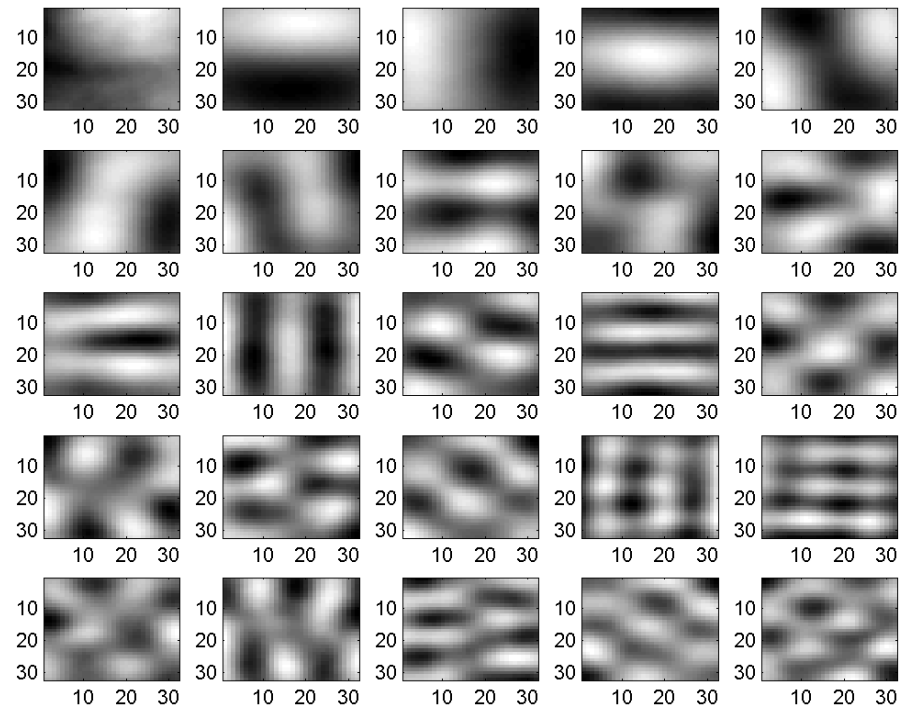


- Composing a fishbowl from Eigenfaces
- With 100 bases

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

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*

Poll 4

Relation of Eigen decomposition to SVD

Eigen Decomposition of the Correlation Matrix

$$\mathbf{XX}^T = \mathbf{R} = \mathbf{E}\mathbf{D}\mathbf{E}^T$$

SVD of the Data Matrix

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

$$\mathbf{XX}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T \mathbf{V}\mathbf{S}\mathbf{U}^T = \mathbf{U}\mathbf{S}^2\mathbf{U}^T$$

Comparing

$$\mathbf{E} = \mathbf{U}$$

$$\mathbf{D} = \mathbf{S}^2$$

- Eigen decomposition of the correlation matrix gives you left singular vectors of data matrix

Dimensionality Reduction

- $\mathbf{R} = \mathbf{E}\mathbf{D}\mathbf{E}^T$
 - The columns of \mathbf{E} are our “Eigen” bases
- We can express any vector X as a combination of these bases

$$X = w_D^X E_1 + w_D^X E_D + \cdots + w_D^X E_D$$

- Using only the “top” K bases
 - Corresponding to the top K Eigen values

$$X \approx w_D^X E_1 + w_D^X E_D + \cdots + w_K^X E_K$$

Dimensionality Reduction

- Using only the “top” K bases
 - Corresponding to the top K Eigen values

$$X \approx w_D^X E_1 + w_D^X E_D + \cdots + w_K^X E_K$$

- In vector form:

$$X \approx \mathbf{E}_{1:K} \mathbf{w}_K^X$$

$$\mathbf{w}_K^X = \text{Pinv}(\mathbf{E}_{1:K})X = \mathbf{E}_{1:K}^T X$$

$$\mathbf{W}_K^X = \mathbf{E}_{1:K}^T \mathbf{X}$$

- If “ \mathbf{E} ” is agreed upon, knowing \mathbf{W}_K^X is sufficient to reconstruct \mathbf{X}
 - Store only K numbers per vector instead of D without losing too much information
 - **Dimensionality Reduction**

Lets give it a name

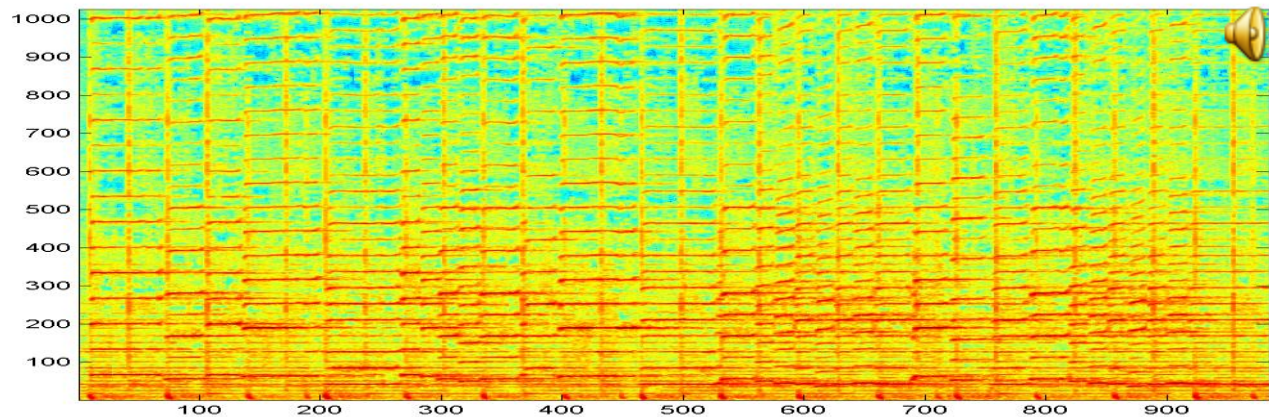
$$\mathbf{R} = \mathbf{E}\mathbf{D}\mathbf{E}^T$$

\mathbf{E} are the “Eigen Bases”

$$\mathbf{W}_K^X = \mathbf{E}_{1:K}^T \mathbf{X}$$

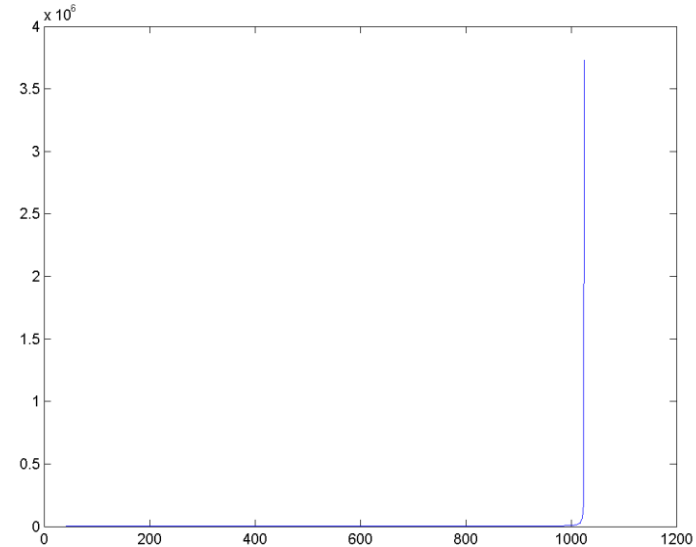
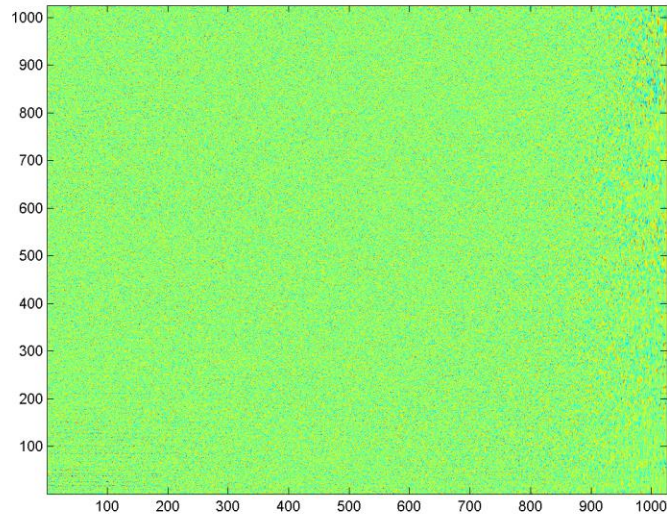
- Retaining only the top K weights for every data vector
 - Computed by multiplying the data matrix by the transpose of the top K Eigen vectors of \mathbf{R}
- This is called the *Karhunen Loeve Transform*
 - *Not PCA!*

An audio example



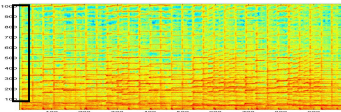
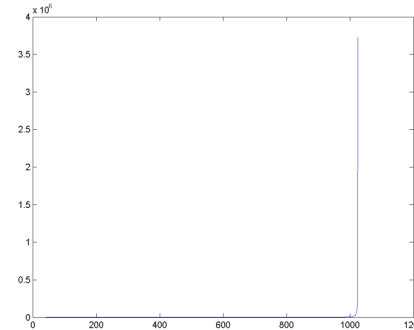
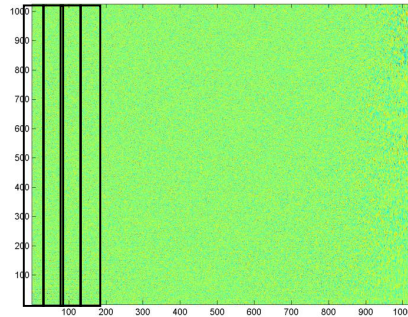
- The spectrogram has 974 vectors of dimension 1025
- The covariance matrix is size 1025 x 1025
- There are 1025 eigenvectors

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

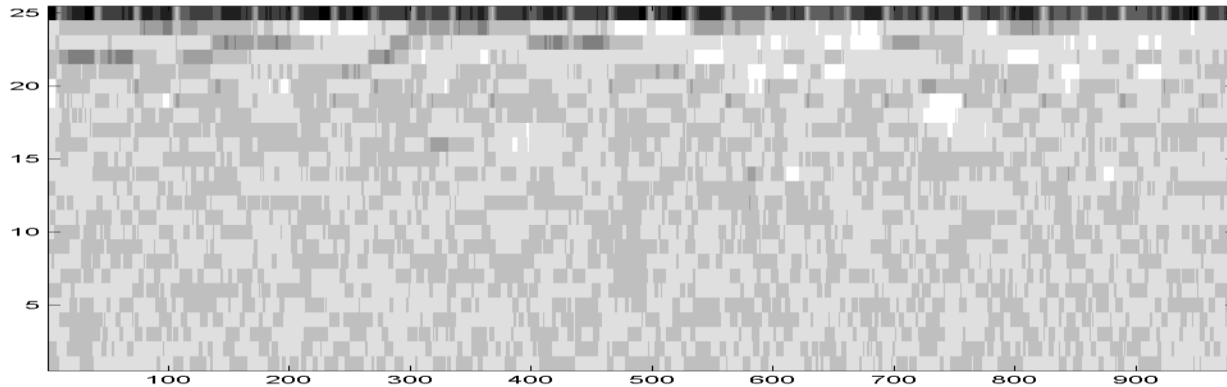


$$\text{Vec} = a_1 * \text{eigenvec1} + a_2 * \text{eigenvec2} + a_3 * \text{eigenvec3} \dots$$

- 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

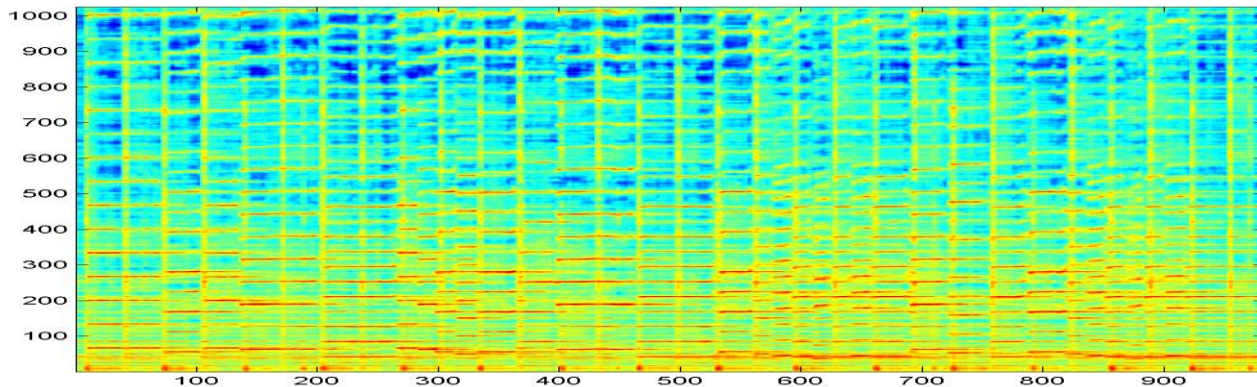
An audio example

$$V_{reduced} = [V_1 \quad \cdot \quad \cdot \quad V_{25}]$$
$$M_{lowdim} = P_{inv}(V_{reduced})M$$



- 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

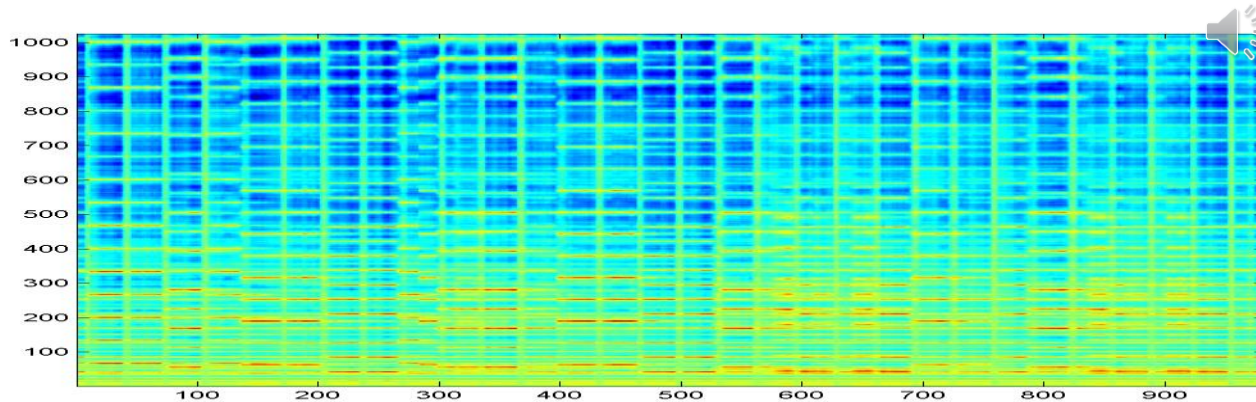
An audio example



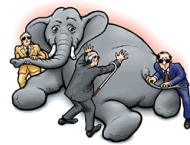
$$M_{reconstructed} = V_{reduced} M_{lowdim}$$

- 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)

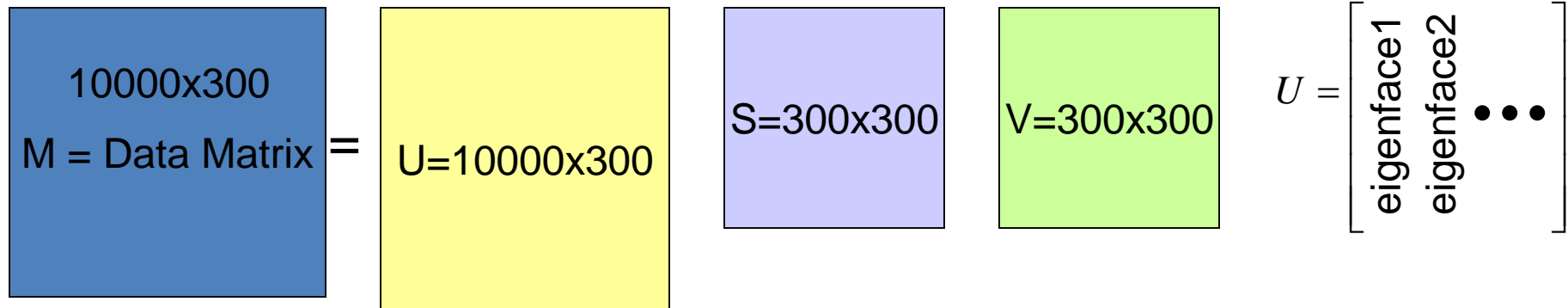
With only 5 eigenvectors



- The same spectrogram constructed from only the 5 Eigen vectors with the highest Eigen values
 - Highly recognizable



SVD instead of Eigen



- 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
 - $U = 10000 \times 300$
 - The columns of U are the eigen faces!
 - The U s corresponding to the “zero” eigen values are not computed
 - $S = 300 \times 300$
 - $V = 300 \times 300$

Using SVD to compute Eigenbases

$$[U, S, V] = \text{SVD}(X)$$

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

$$[U, S, V] = \text{svds}(X, 100)$$

- 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

What kind of representation?

- What we just saw: *Karhunen Loeve Expansion*
- What you may be familiar with: *Principal Component Analysis*
- The two are similar, but not the same!!

Linear vs. Affine

- The model we saw (KLE)

- Approximate **every** face f as

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

- Linear combination of bases

- If you add a constant (PCA)

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

- *Affine* combination of bases

Affine expansion

- Estimate

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

- Using the *energy compaction* principle leads to the usual incremental estimation rule
 - m must explain most of the energy
 - Each new basis must explain most of the residual energy

Estimation with the constant

- Estimate

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

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

Estimation with the constant

- Estimate

$$f \approx m$$

– 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:
 - $\hat{f} = w_{f,1} V_1 + w_{f,2} V_2 + \dots + w_{f,k} V_k$
- This is just Eigen analysis of the “mean-normalized” data
 - Also called the “centered” data

Estimating the Affine model

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

- First estimate the mean m

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

- Compute the correlation matrix of the “centered” data $\hat{f} = f - m$

$$- \mathbf{C} = \sum_f \hat{f} \hat{f}^T = \sum_f (f - m)(f - m)^T$$

- This is the *covariance* matrix of the set of f

Estimating the Affine model

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

- First estimate the mean m

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

- Compute the covariance matrix

$$- C = \sum_f (f - m)(f - m)^T$$

- Eigen decompose!

$$CV = V\Lambda$$

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

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 Expansion***

- 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

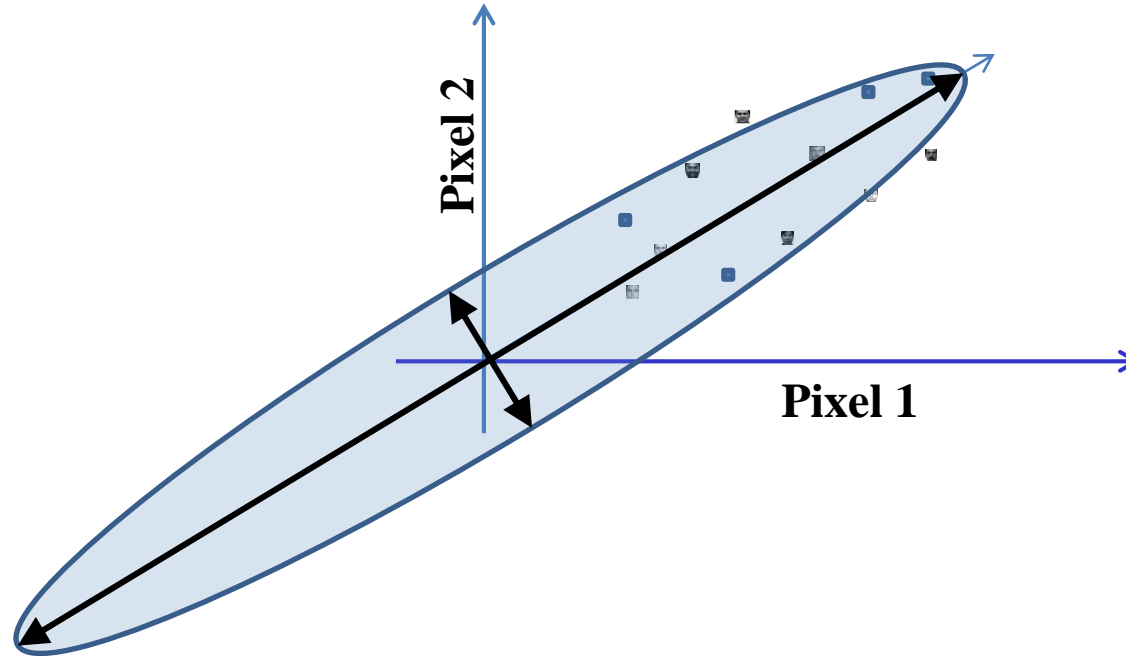
How do they relate

- Relationship between correlation matrix and covariance matrix

$$\mathbf{R} = \mathbf{C} + \mathbf{m}\mathbf{m}^T$$

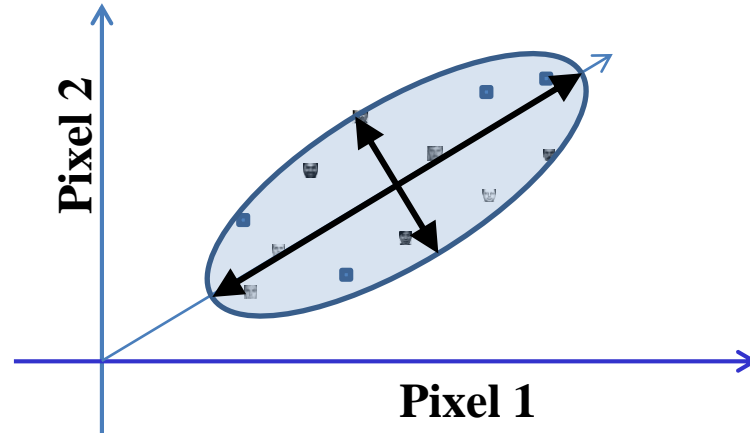
- *Karhunen Loeve* bases are Eigen vectors of \mathbf{R}
- *PCA* bases are Eigen vectors of \mathbf{C}
- How do they relate
 - Not easy to say..

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

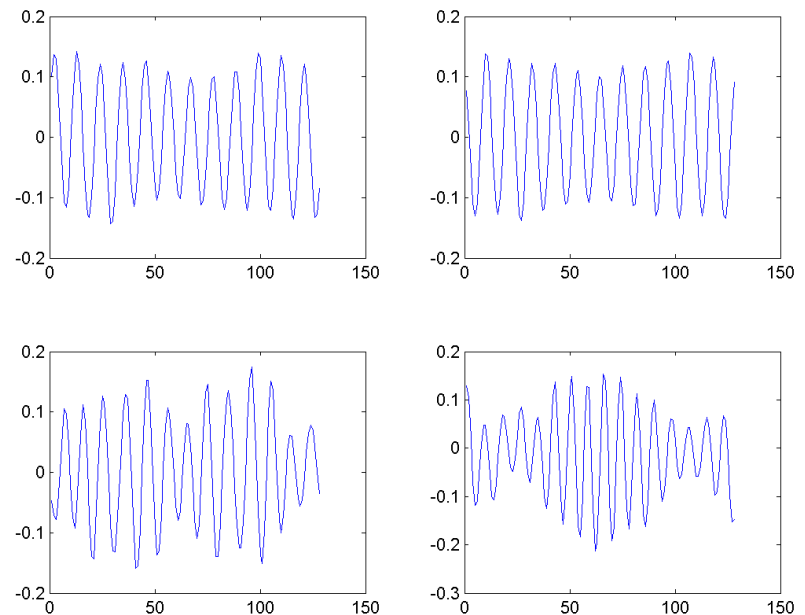


- Turns out: Eigenvectors of the *covariance* represent the major and minor axes of an ellipse centered at the *mean* which encloses the data most compactly
- PCA uncovers these vectors
- In practice, “Eigen faces” refers to *PCA* faces, and not KLT faces

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



Figure 1. 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.

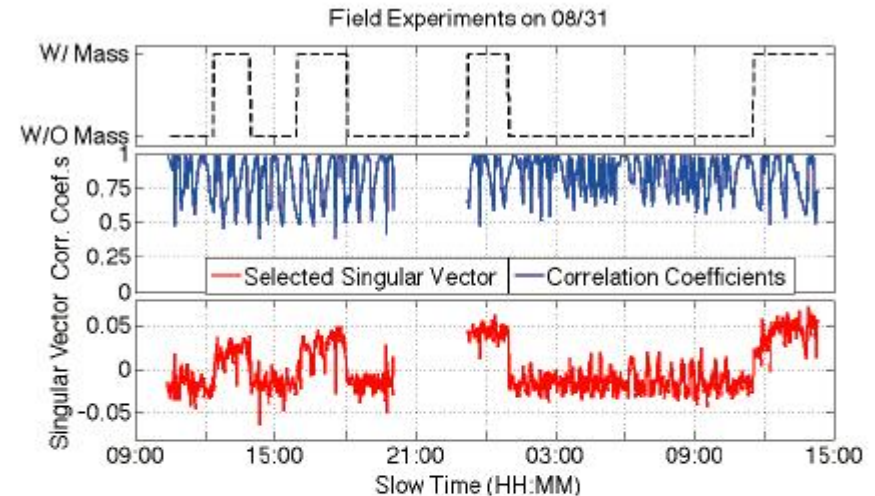


Figure 2. Damage detection results compared with conventional methods. **Top:** Ground truth of whether the pipe is damaged or not. **Middle:** Conventional method only captures temperature variations, and shows no indication of the presence of damage. **Bottom:** The SVD method clearly picks up the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

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