

# CPSC 340: Machine Learning and Data Mining

More PCA

Fall 2021

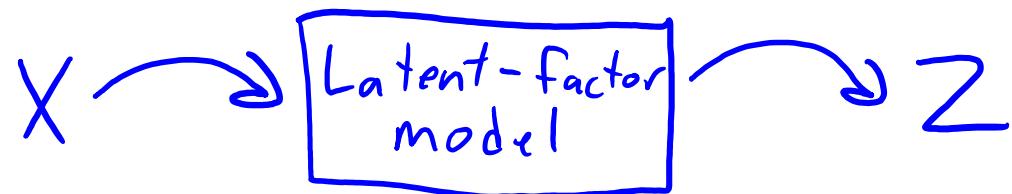
1. Decision trees
2. Naïve Bayes classification
3. Ordinary least squares regression
4. Logistic regression
5. Support vector machines
6. Ensemble methods
7. Clustering algorithms
8. Principal component analysis
9. Singular value decomposition
10. Independent component analysis (bonus)

## The 10 Algorithms Machine Learning Engineers Need to Know

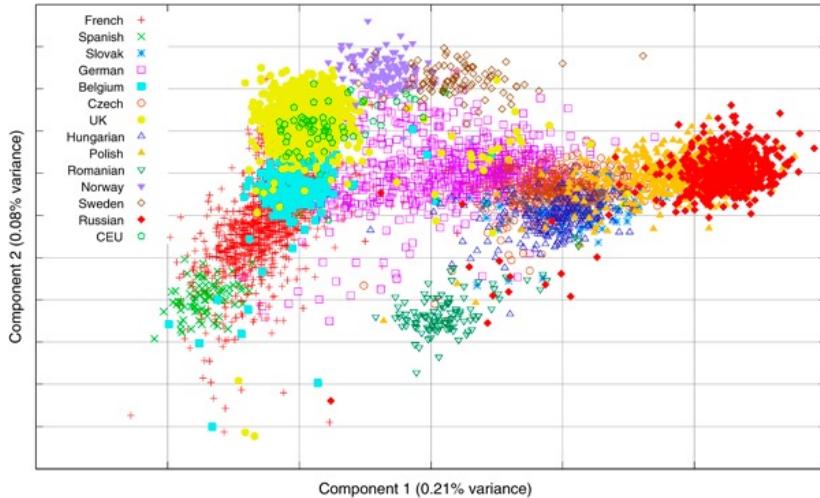


# Last Time: Latent-Factor Models

- Latent-factor models take input data 'X' and output a basis 'Z':



- Usually, 'Z' has fewer features than 'X'.
- Uses: dimensionality reduction, visualization, factor discovery.



Trait	Description
Openness	Being curious, original, intellectual, creative, and open to new ideas.
Conscientiousness	Being organized, systematic, punctual, achievement-oriented, and dependable.
Extraversion	Being outgoing, talkative, sociable, and enjoying social situations.
Agreeableness	Being affable, tolerant, sensitive, trusting, kind, and warm.
Neuroticism	Being anxious, irritable, temperamental, and moody.

<http://infoproc.blogspot.ca/2008/11/european-genetic-substructure.html>

<https://new.edu/resources/big-5-personality-trait>

# Last Time: Principal Component Analysis

- Principal component analysis (PCA) is a linear latent-factor model:
  - These models “factorize” matrix  $X$  into matrices  $Z$  and  $W$ :

$$X \approx ZW$$

$n \times d$

$$x_i \approx W^T z_i$$

$n \times k$

$$x_{ij} \approx \langle w_j^T, z_i \rangle$$

$k \times d$

- We can think of rows  $w_c$  of  $W$  as ‘ $k$ ’ fixed “part” (used in all examples).
- $z_i$  is the “part weights” for example  $x_i$ : “how much of each part  $w_c$  to use”.

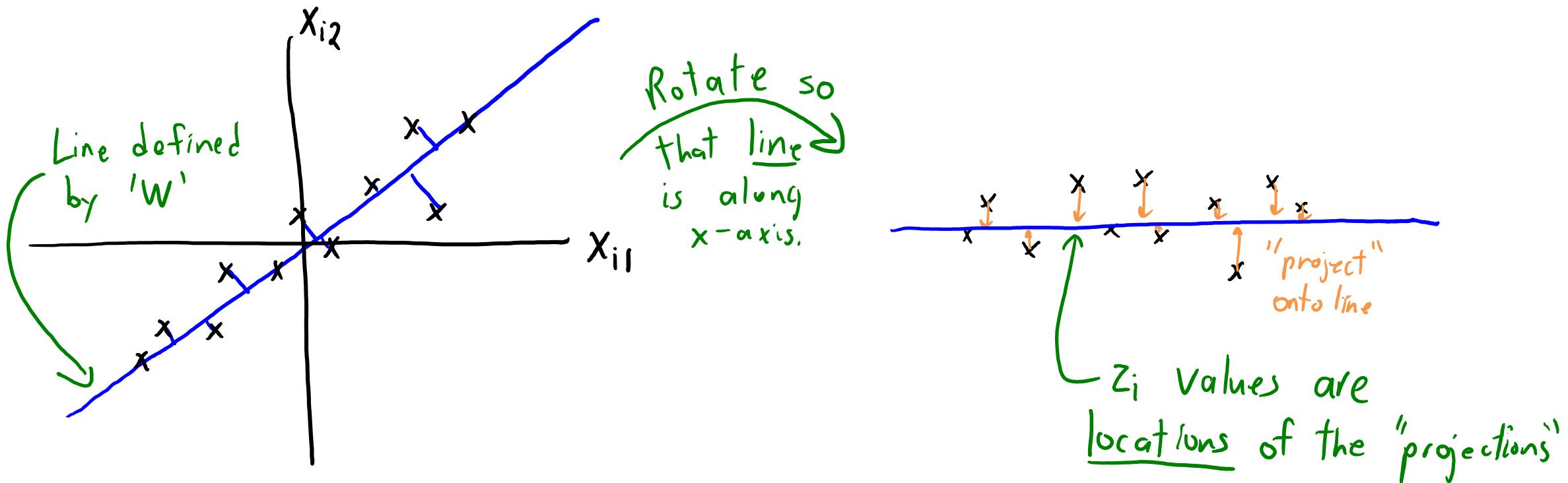
$$\begin{matrix} 5 \\ = 1 & + 0 & + 1 & + 1 & + 1 & + 0 & + 1 \end{matrix}$$

$\hat{x}_i = z_{i1} w_1 + z_{i2} w_2 + z_{i3} w_3 + z_{i4} w_4 + z_{i5} w_5 + z_{i6} w_6 + z_{i7} w_7$

“parts” are called “principal components” or “factors”

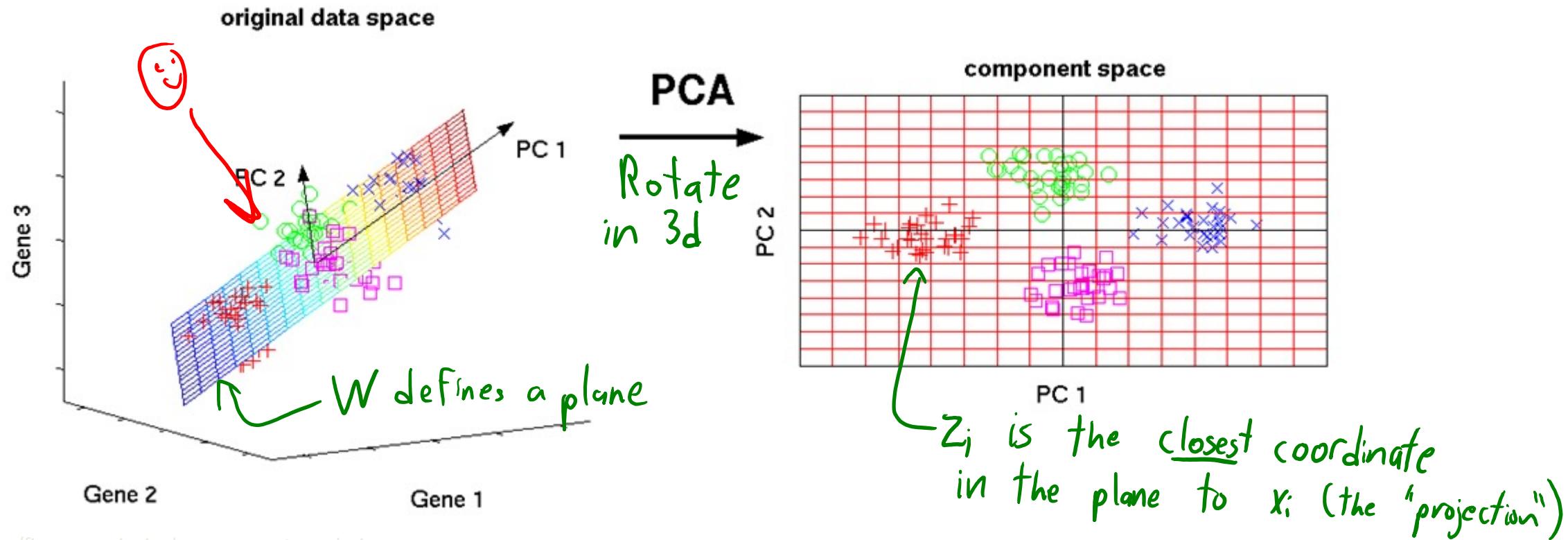
# Last Time: PCA Geometry

- When  $k=1$ , the  $W$  matrix defines a line:
  - We choose ' $W$ ' as the line minimizing squared distance to the data.
  - Given ' $W$ ', the  $z_i$  are the coordinates of the  $x_i$  “projected” onto the line.



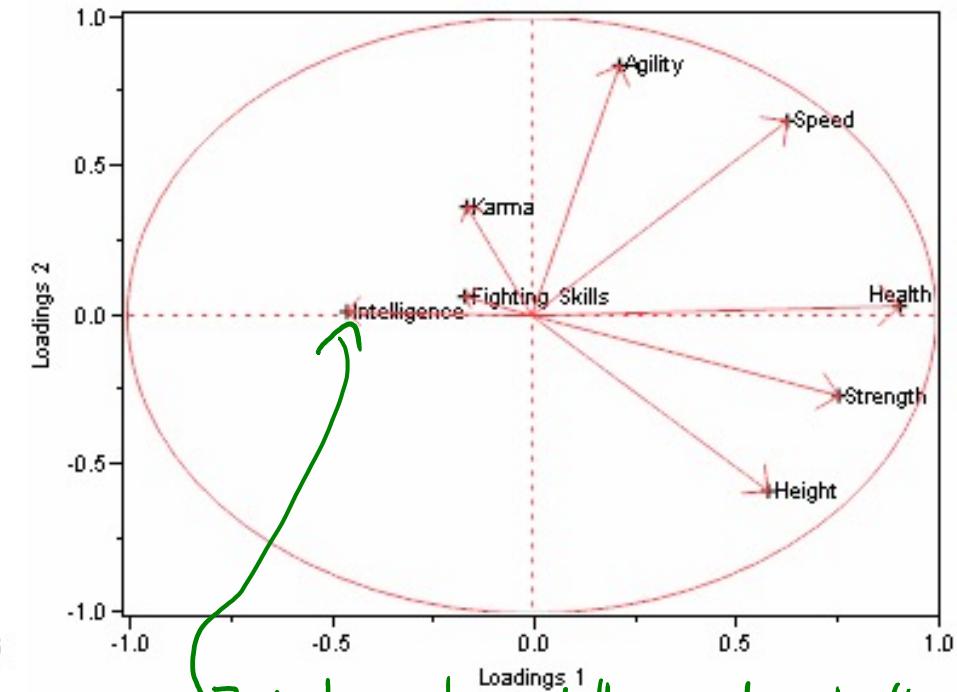
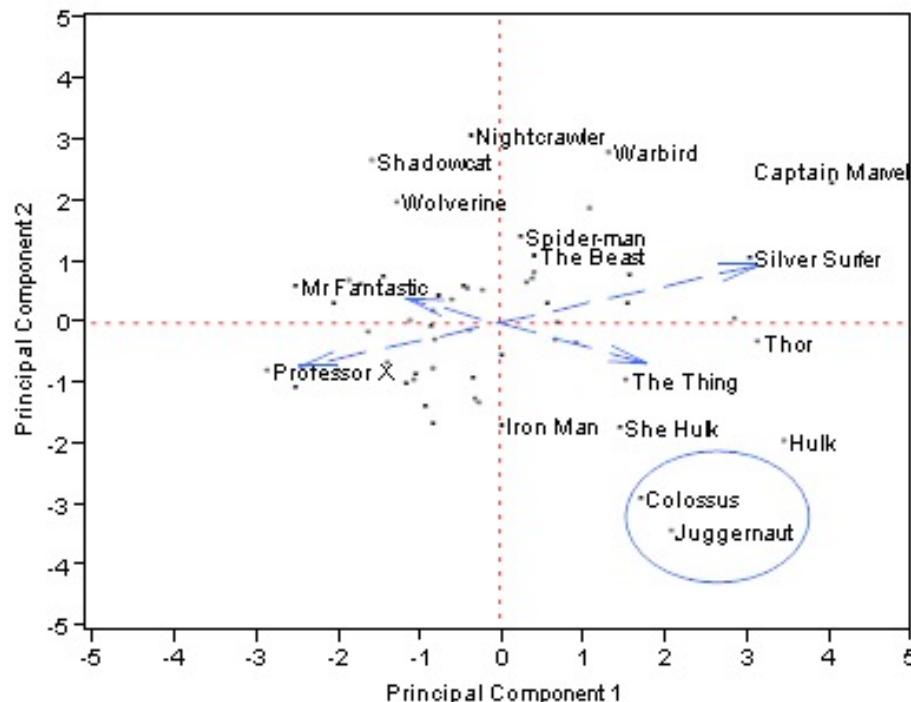
# Last Time: PCA Geometry

- When  $k=2$ , the  $W$  matrix defines a **plane**:
  - We choose ‘ $W$ ’ as the **plane minimizing squared distance to the data**.
  - Given ‘ $W$ ’, the  $z_i$  are the coordinates of the  $x_i$  “projected” onto the plane.



# Last Time: PCA Geometry

- When  $k=2$ , the  $W$  matrix defines a **plane**:
  - Even if the original data is high-dimensional,  
we can **visualize** data “projected” onto this plane.



# PCA Objective Function

- In PCA we minimize the squared error of the approximation:

$$f(W, z) = \sum_{i=1}^n \|W^T z_i - \underbrace{x_i}_{\text{approximation example } i}\|^2$$

- This is equivalent to the k-means objective:
  - In k-means  $z_i$  only has a single '1' value and other entries are zero.
- But in PCA,  $z_i$  can be any real number.
  - We approximate  $x_i$  as a linear combination of all means/factors.

# PCA Objective Function

- In PCA we minimize the squared error of the approximation:

$$f(W, z) = \sum_{i=1}^n \|W^T z_i - x_i\|^2 = \sum_{i=1}^n \sum_{j=1}^d (\langle w_j^i, z_i \rangle - x_{ij})^2$$

approximation example 'i'      approximating feature 'j' of example 'i'

- We can also view this as solving 'd' regression problems:
  - Each  $w_j^i$  is trying to predict column ' $x_j$ ' from the basis  $z_i$ .
    - The output " $y_i$ " we try to predict here is actually the features " $x_i$ ".
  - And unlike in regression we are also learning the features  $z_i$ .

# Principal Component Analysis (PCA)

- The 3 different ways to write the PCA objective function:

$$\begin{aligned} f(W, Z) &= \sum_{i=1}^n \sum_{j=1}^d (\langle w_j^T z_i \rangle - x_{ij})^2 && \text{(approximating } x_{ij} \text{ by } \langle w_j^T z_i \rangle \text{)} \\ &= \sum_{i=1}^n \|W^T z_i - x_i\|^2 && \text{(approximating } x_i \text{ by } W^T z_i \text{)} \\ &= \|Z^T W - X\|_F^2 && \text{(approximating } X \text{ by } Z^T W \text{)} \end{aligned}$$

# Digression: Data Centering (Important)

- In PCA, we assume that the data  $X$  is “centered”.
  - Each column of  $X$  has a mean of zero.
- It's easy to center the data:

Set  $\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$  (mean of column ' $j$ ')

Replace each  $x_{ij}$  with  $(x_{ij} - \mu_j)$

- There are PCA variations that estimate “bias in each coordinate”.
  - In basic model this is equivalent to centering the data.

# PCA Computation: Prediction

- At the end of training, the “model” is the  $\mu_j$  and the  $W$  matrix.
  - PCA is parametric.
- PCA prediction phase:
  - Given new data  $\tilde{X}$ , we can use  $\mu_j$  and  $W$  this to form  $\tilde{Z}$ :

1. Center: replace each  $\tilde{x}_{ij}$  with  $(\tilde{x}_{ij} - \mu_j)$

2. Find  $\tilde{Z}$  minimizing squared error:

$$\tilde{Z} = \tilde{X} W^T (W W^T)^{-1}$$

(could just store  
this  $d \times k$  matrix)

$\overbrace{\mu_j}$  means of  
training  
data

# PCA Computation: Prediction

- At the end of training, the “model” is the  $\mu_j$  and the  $W$  matrix.
  - PCA is parametric.
- PCA prediction phase:
  - Given new data  $\tilde{X}$ , we can use  $\mu_j$  and  $W$  this to form  $\tilde{Z}$ :
  - The “reconstruction error” is how close approximation is to  $\tilde{X}$ :

$$\|\underbrace{\tilde{Z}W}_{\hat{X}} - \underbrace{\tilde{X}}_{\text{Centered version}}\|_F^2$$

- Our “error” from replacing the  $x_i$  with the  $z_i$  and  $W$ .

# Choosing ‘k’ by “Variance Explained”

- Common to choose ‘k’ based on variance of the  $x_{ij}$ .

$$\text{Var}(x_{ij}) = E[(x_{ij} - \mu_{ij})^2] = E[x_{ij}^2] = \frac{1}{nd} \sum_{i=1}^n \sum_{j=1}^d x_{ij}^2 = \frac{1}{nd} \|X\|_F^2$$

definition of variance  
assumed to be zero  
definition of expectation  
Frobenius norm

- For a given ‘k’ we compute (variance of errors)/(variance of  $x_{ij}$ ):

$$\frac{\|ZW - X\|_F^2}{\|X\|_F^2}$$

Centred version

- Gives a number between 0 ( $k=d$ ) and 1 ( $k=0$ ), giving “variance remaining”.
  - If you want to “explain 90% of variance”, choose smallest ‘k’ where ratio is < 0.10.

# “Variance Explained” in the Doom Map

- Recall the Doom latent-factor model (where map ignores height):



- Interpretation of “variance remaining” formula:

$$\frac{\|Z_W - X\|_F^2}{\|X\|_F^2} \leftarrow \begin{array}{l} \text{Variance in } z\text{-dimension (variance in } x\text{- and } y\text{-dimensions fully} \\ \text{captured by overhead map)} \end{array}$$

$\leftarrow$  Variance of character in 3-dimensions

- If we had a 3D map the “variance remaining” would be 0.

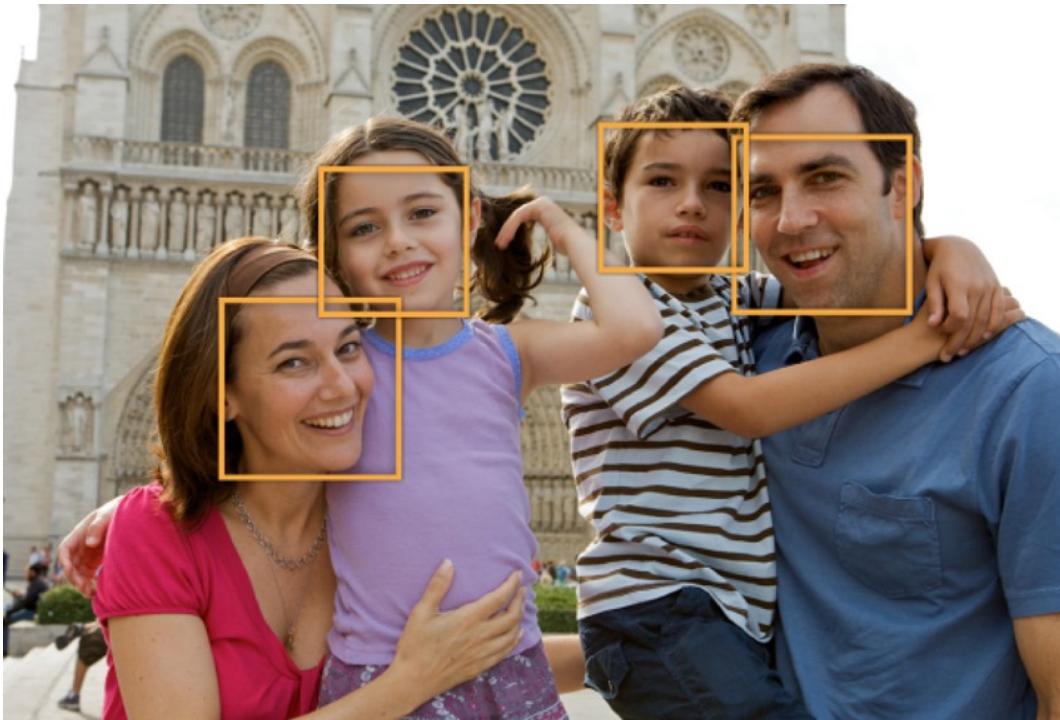
[https://en.wikipedia.org/wiki/Doom\\_\(1993\\_video\\_game\)](https://en.wikipedia.org/wiki/Doom_(1993_video_game))

<https://forum.minetest.net/viewtopic.php?f=5&t=9666>

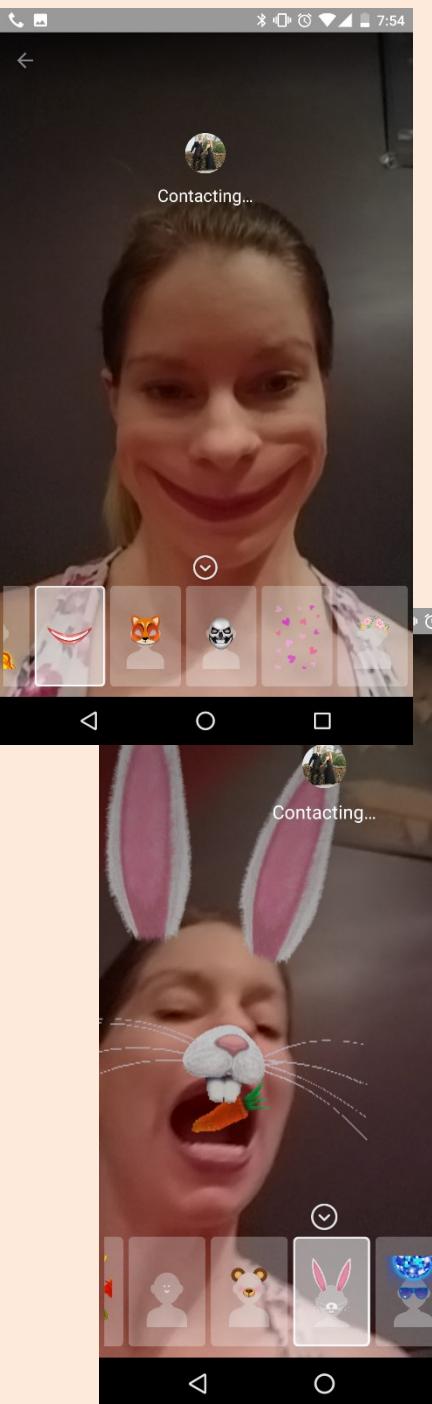
(pause)

# Application: Face Detection

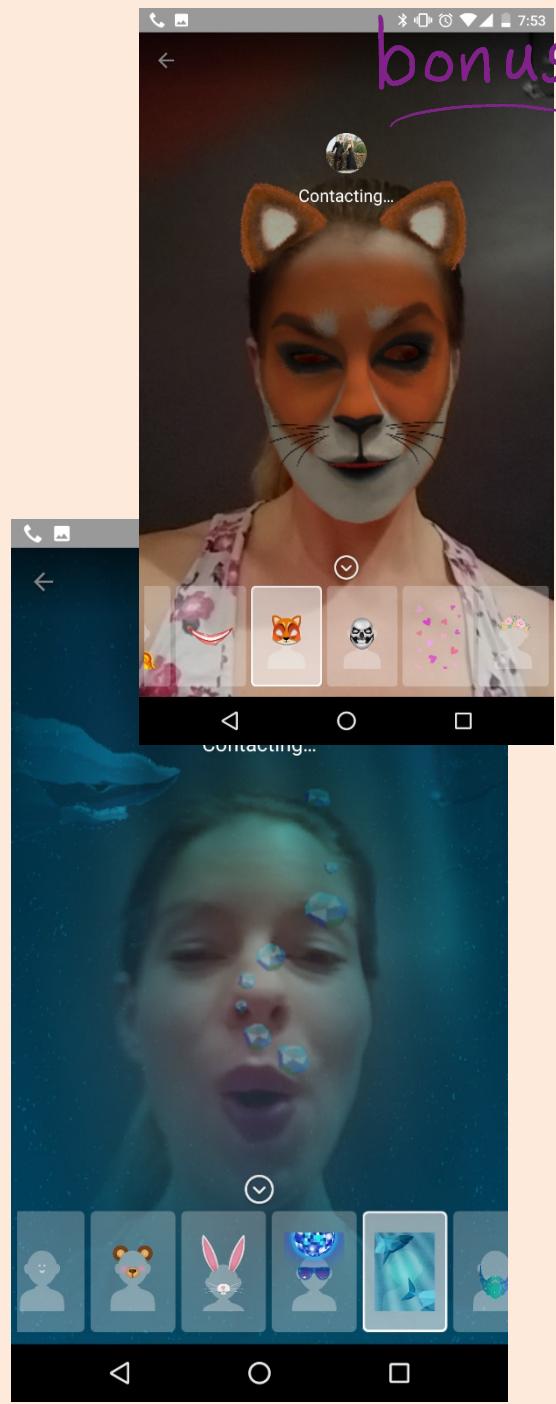
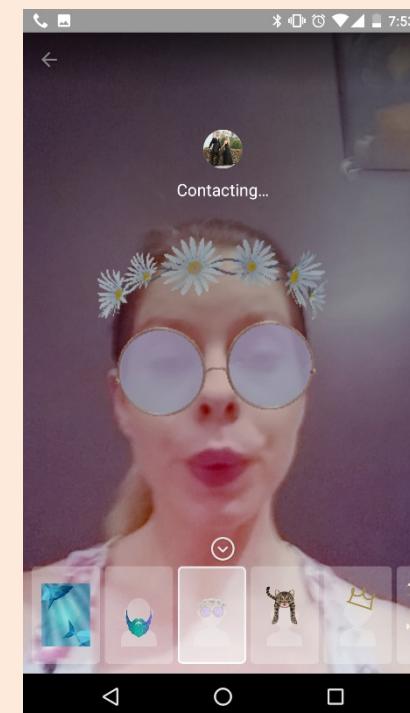
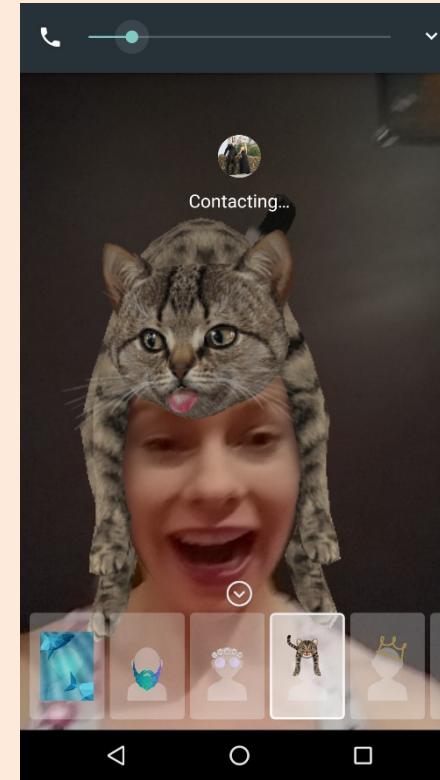
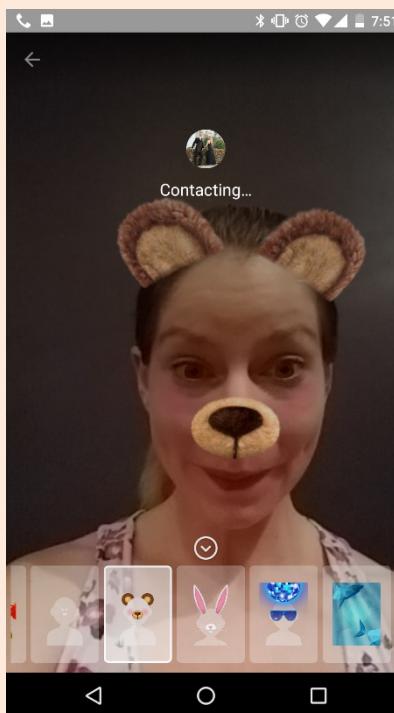
- Consider problem of face detection:



- Classic methods use “eigenfaces” as basis:
  - PCA applied to images of faces.



# Application: Face Detection



# Eigenfaces

- Collect a bunch of images of faces under different conditions:



Each row of  $X$  will be pixels in one image:

$$X =$$

If have ' $n$ ' images that are ' $m$ ' by ' $m$ ' then  $X$  is ' $n$ ' by  $m^2$ .

# Eigenfaces

Compute mean  $\mu_j$  of each column.



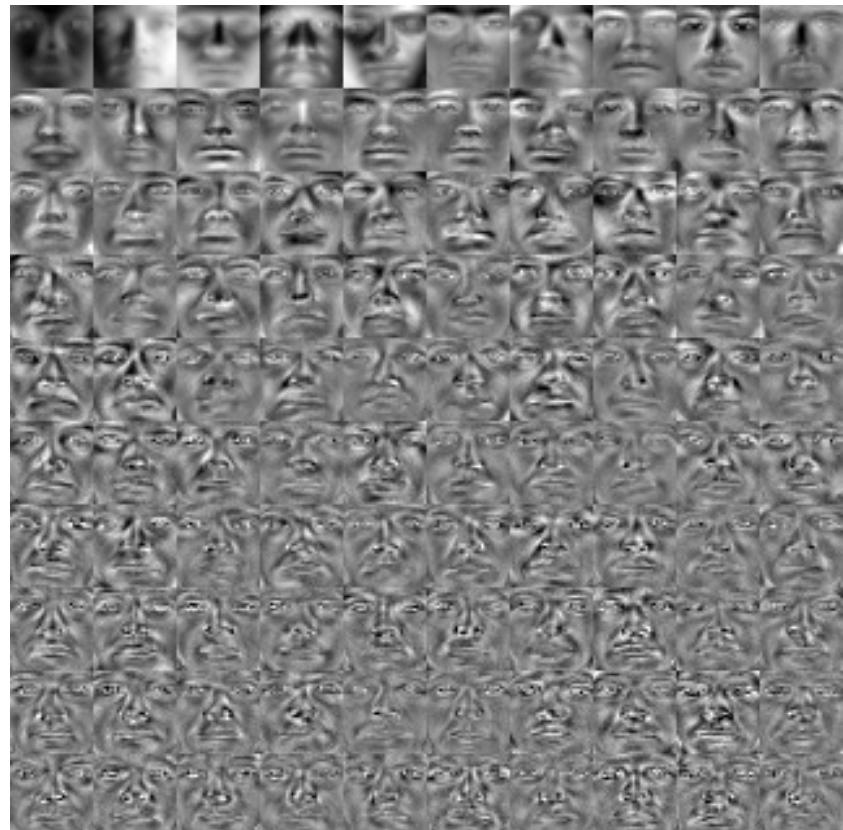
Each row of  $X$  will be pixels in one image:

$$X = \begin{bmatrix} x_1 - \mu \\ x_2 - \mu \\ \vdots \\ x_n - \mu \end{bmatrix}$$

Replace each  $x_{ij}$  by  $x_{ij} - \mu_j$

# Eigenfaces

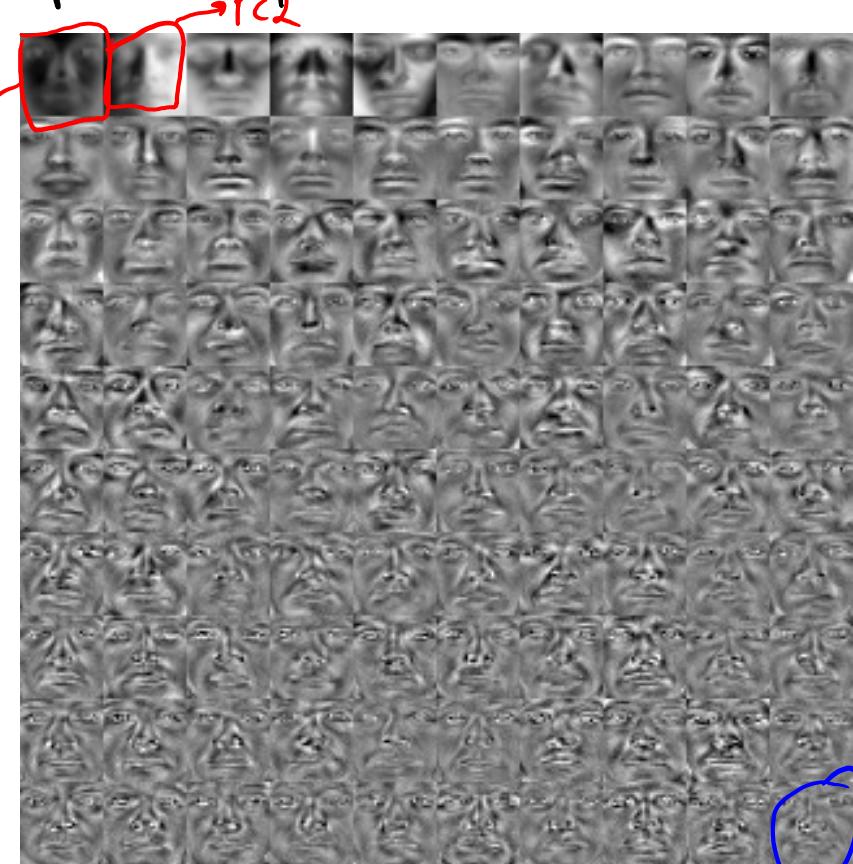
Compute top 'k' PCs on centered data: Each row of  $X$  will be pixels in one image:



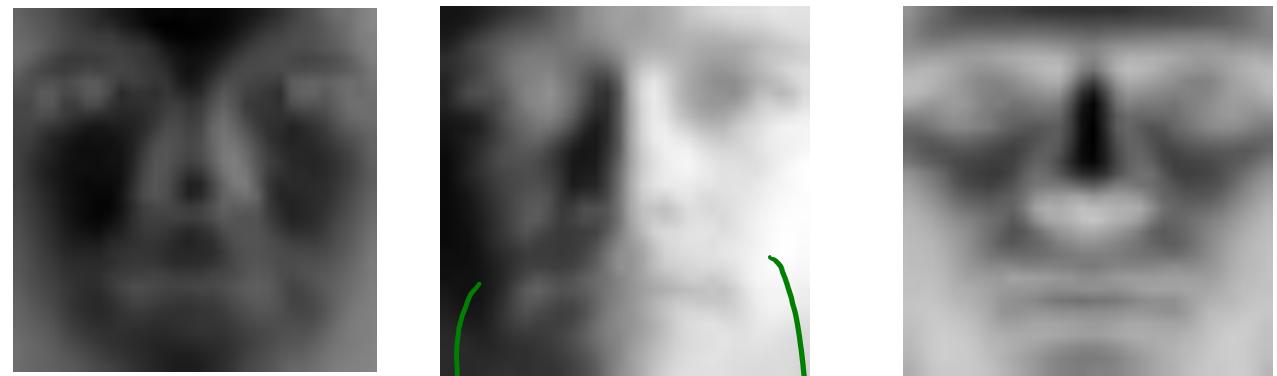
$$X = \begin{bmatrix} x_1 - \mu \\ x_2 - \mu \\ \vdots \\ x_n - \mu \end{bmatrix}$$

# Eigenfaces

Compute top 'k' PCs on centered data:



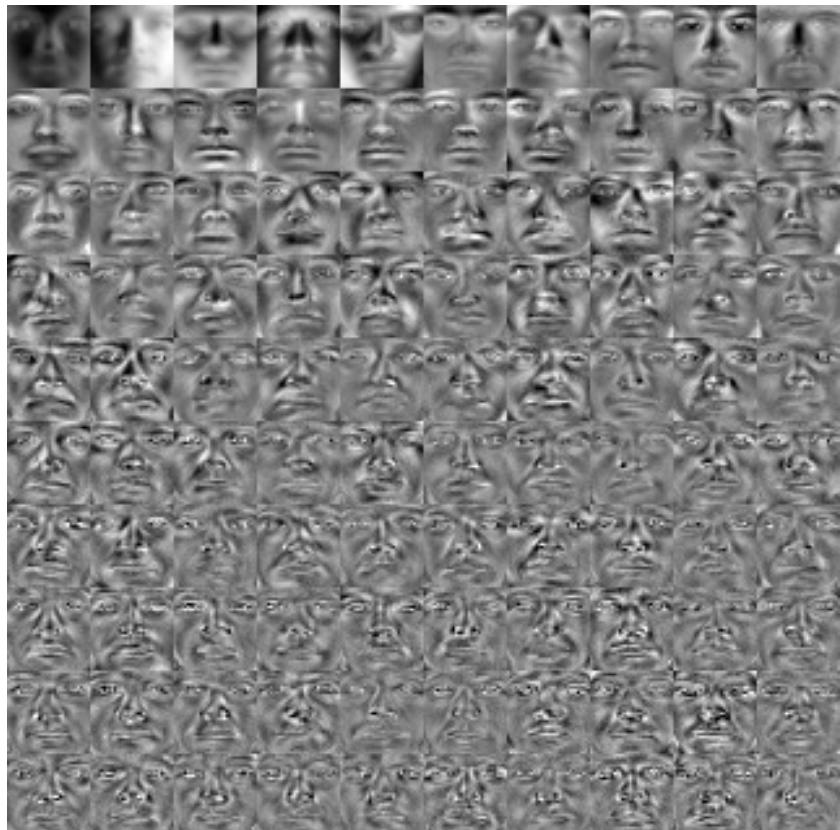
Note that these are "signed" images.



"gray" represents values close to 0. "dark" represents negative values "bright" represents positive values

# Eigenfaces

Compute top 'k' PCs on centered data:



"Eigenface" representation:

$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

$\hat{x}_i$        $\mu$        $z_{i1}$        $\text{PC1}$   
 $(\text{first row of } W)$        $z_{i2}$        $\text{PC2}$        $z_{i3}$        $\text{PC3}$        $\dots$

# Eigenfaces

106 of the original faces:



"Eigenface" representation:

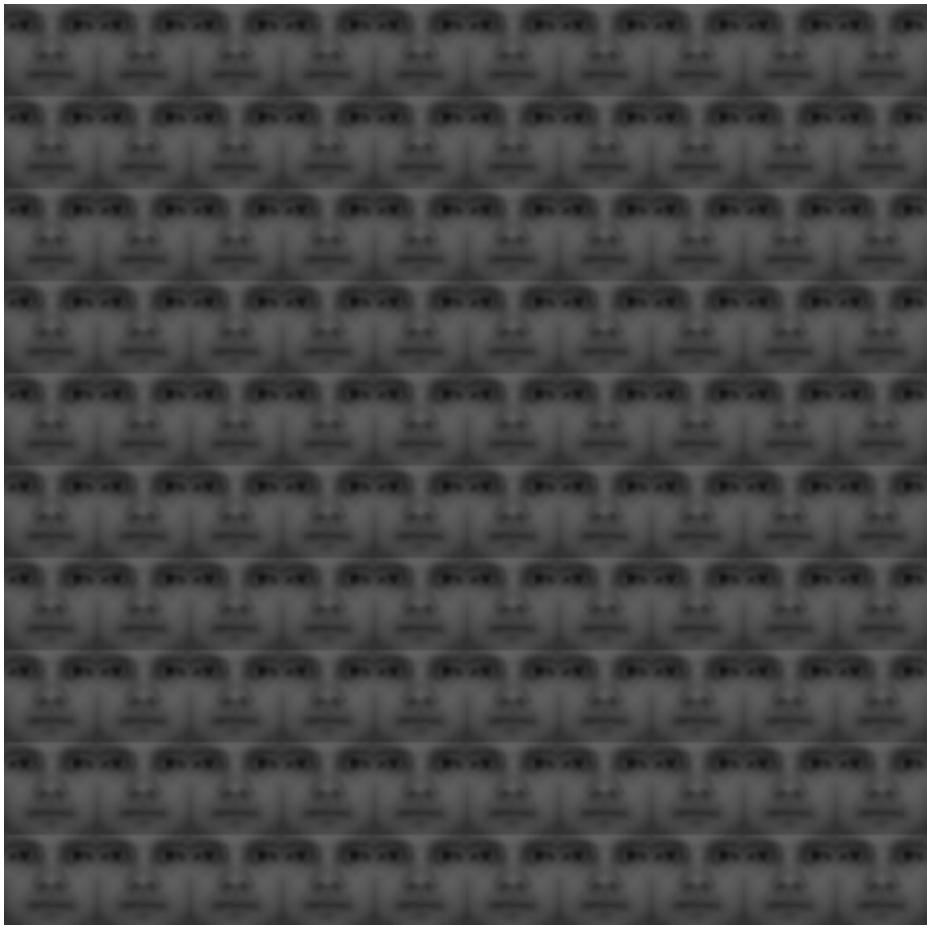
$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

$\hat{x}_i$        $\mu$        $z_{i1}$        $\text{PC1}$   
                         $z_{i2}$        $\text{PC2}$   
                         $z_{i3}$        $\text{PC3}$   
                         $\vdots$

(first row of  $W$ )

# Eigenfaces

Reconstruction with  $k=0$



Variance explained: 0%

"Eigenface" representation:

$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

$\hat{x}_i$   $\mu$   $z_{i1}$   $z_{i2}$   $z_{i3}$   $\text{PC1}$   $\text{PC2}$   $\text{PC3}$

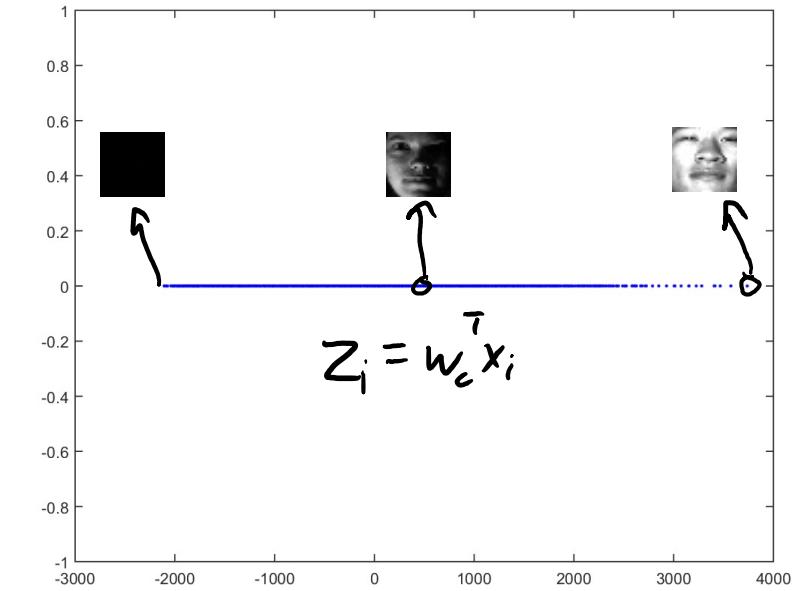
(first row of  $W$ )

# Eigenfaces

Reconstruction with  $k=1$



PCA Visualization:



"Eigenface" representation:

Variance explained: 36%

$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

The equation shows the decomposition of a reconstructed face  $\hat{x}_i$  into its mean  $\mu$  and its projections onto the first three principal components (Eigenfaces)  $\text{PC1}$ ,  $\text{PC2}$ , and  $\text{PC3}$ . A red bracket groups the terms  $\mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3}$ . Another red bracket groups the term  $(\text{first row of } w)$  under the label  $\text{PC1}$ .

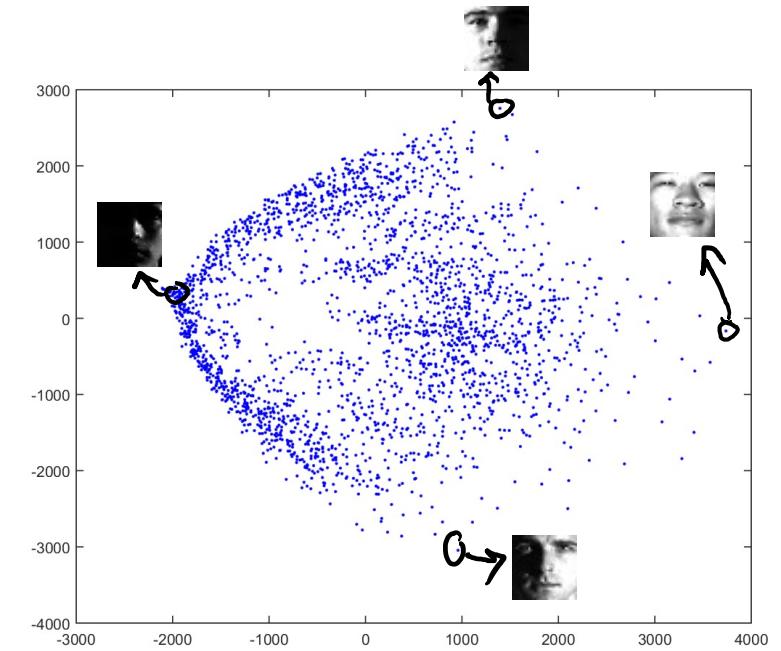
# Eigenfaces

Reconstruction with  $k=2$



Variance explained: 71%

PCA Visualization:



"Eigenface" representation:

$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

The equation shows the "Eigenface" representation of a face  $\hat{x}_i$ . It is represented as the sum of the mean face  $\mu$ , the first principal component  $\text{PC1}$  scaled by the coefficient  $z_{i1}$ , the second principal component  $\text{PC2}$  scaled by  $z_{i2}$ , the third principal component  $\text{PC3}$  scaled by  $z_{i3}$ , and so on. A red circle highlights the term  $\hat{x}_i$ .

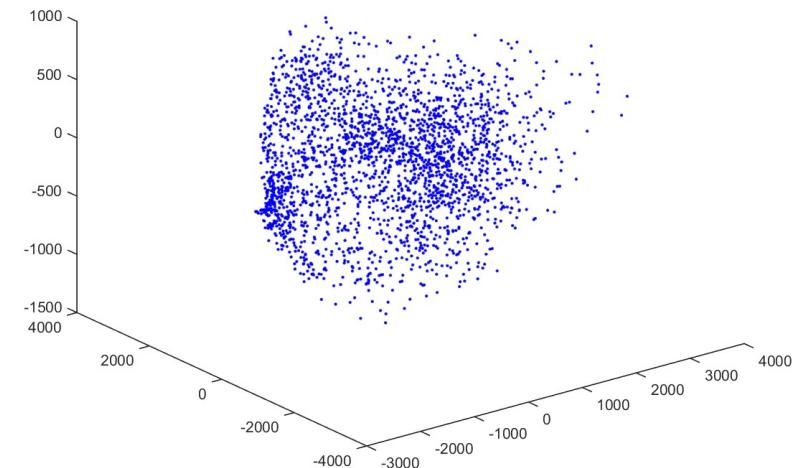
# Eigenfaces

Reconstruction with  $k=3$



Variance explained: 76%

PCA Visualization:



"Eigenface" representation:

$$\hat{x}_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

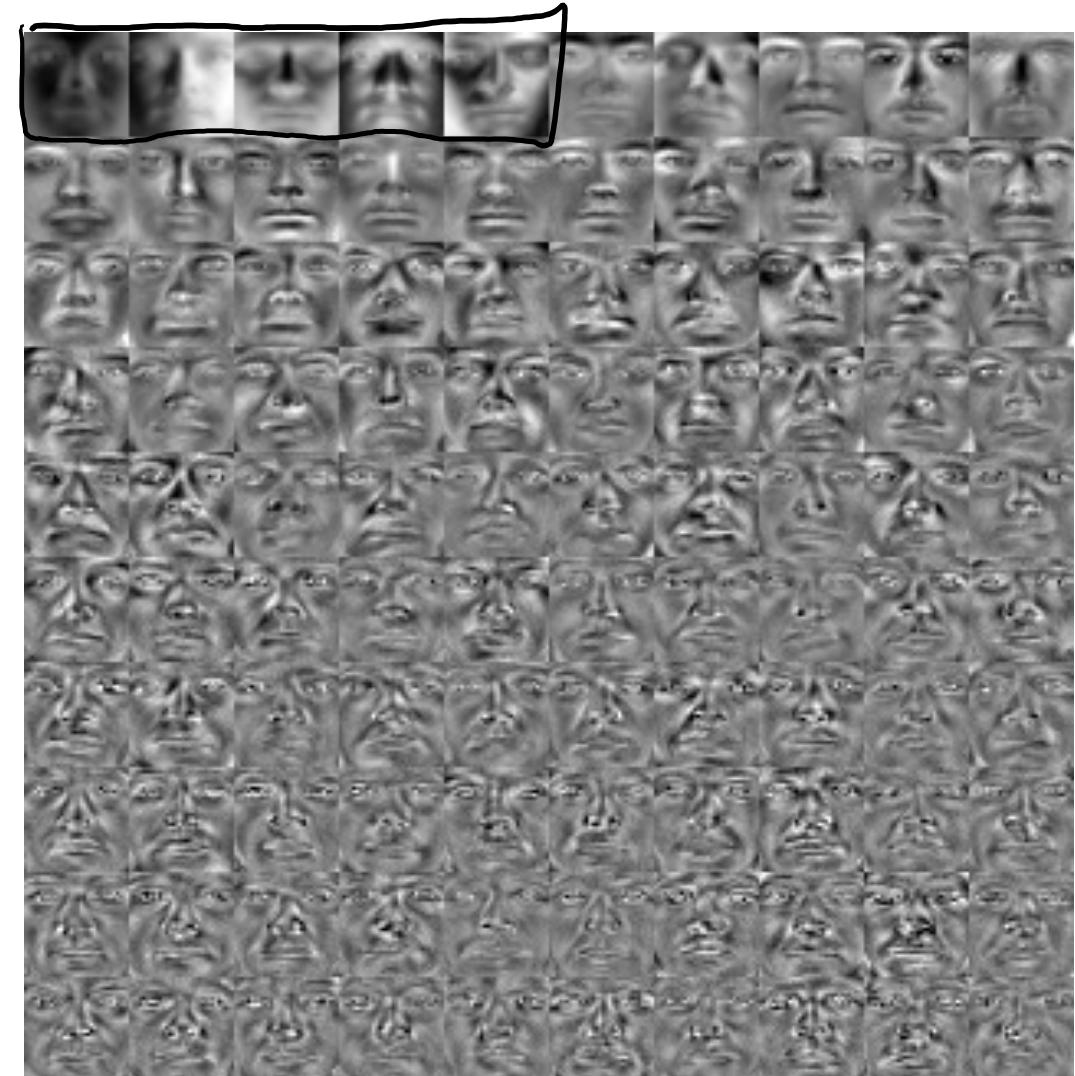
The equation shows the "Eigenface" representation of a face  $\hat{x}_i$ . It is expressed as the sum of the mean face  $\mu$ , three principal components (eigenfaces)  $\text{PC1}, \text{PC2}, \text{PC3}$ , and a residual term  $+ \dots$ . The first row of the matrix  $W$  is highlighted with a red box around the terms  $\text{PC1}, \text{PC2}, \text{PC3}$ .

# Eigenfaces

Reconstruction with  $k=5$



Variance explained: 80 %

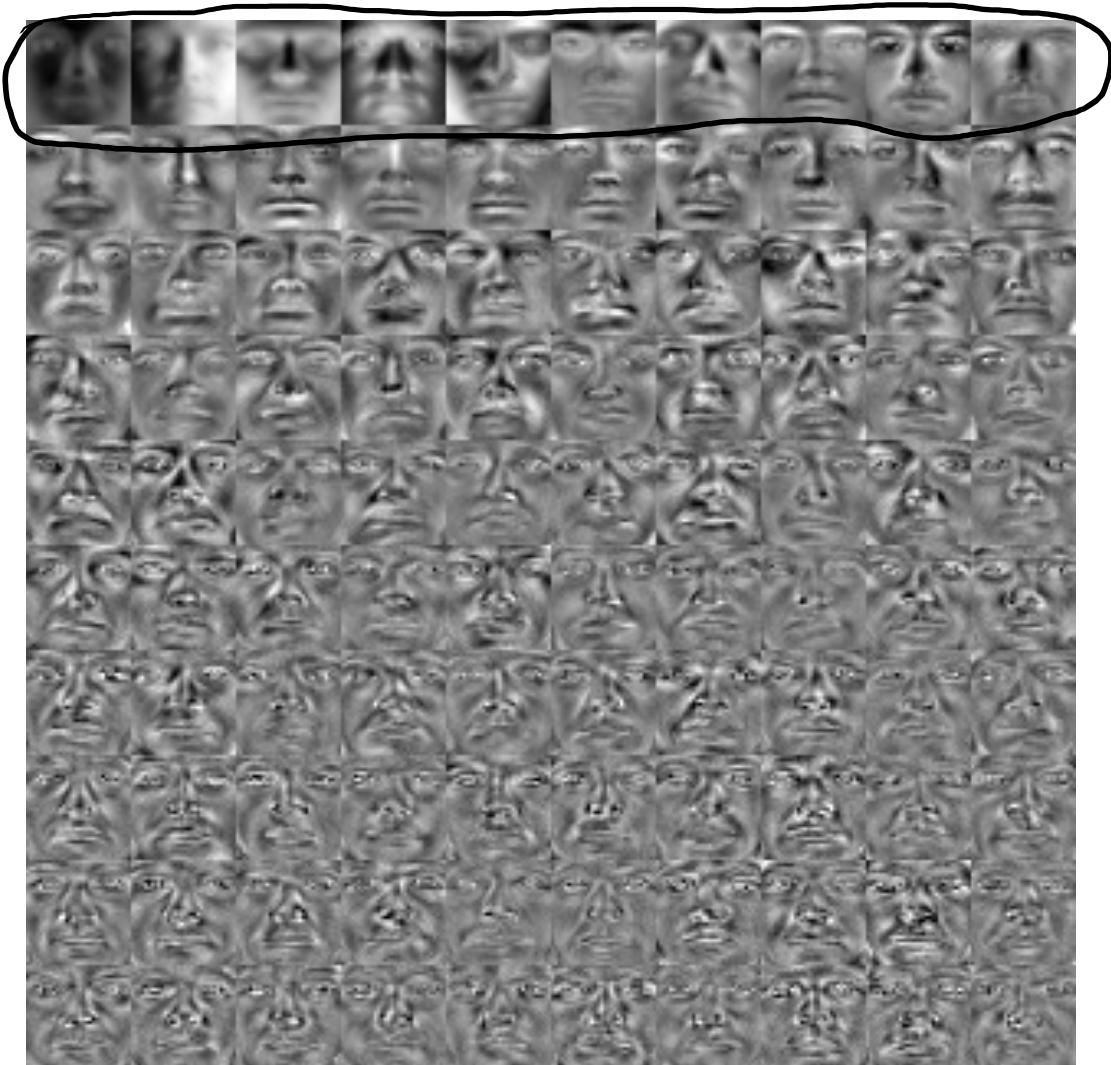


# Eigenfaces

Reconstruction with  $k=10$

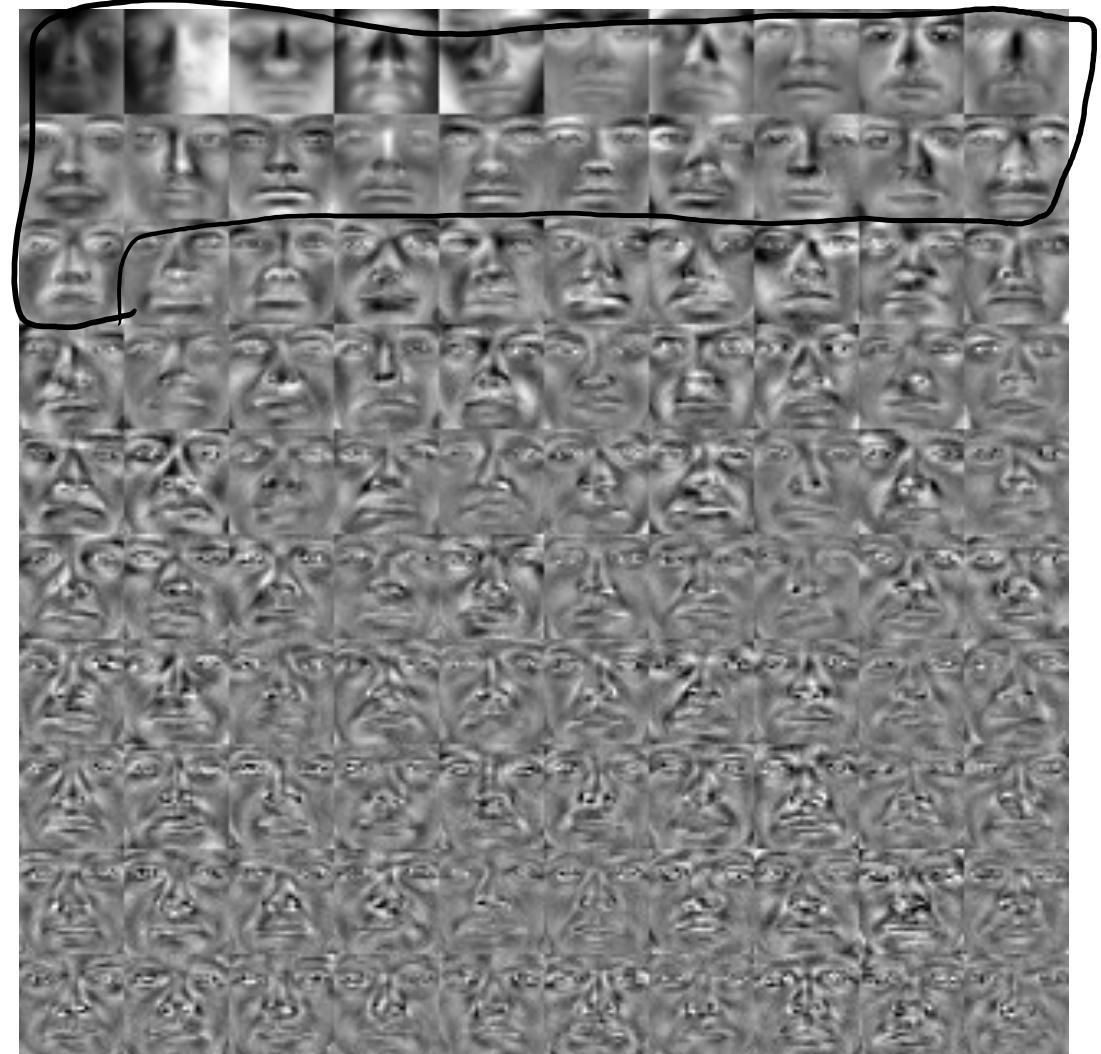


Variance explained: 85%



# Eigenfaces

Reconstruction with  $k=21$



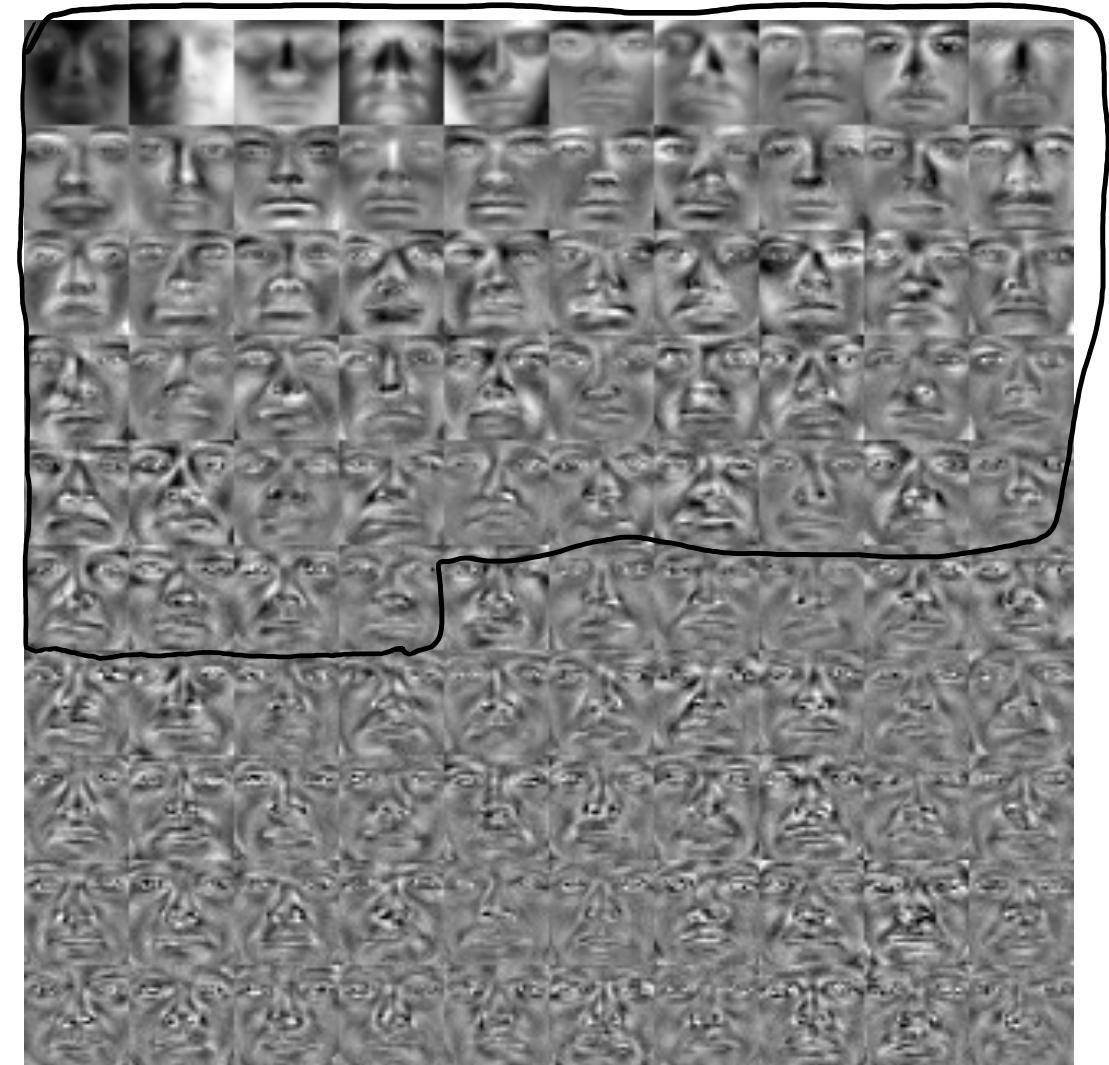
Variance explained: 90%

# Eigenfaces

Reconstruction with  $k=54$



Variance explained: 95%

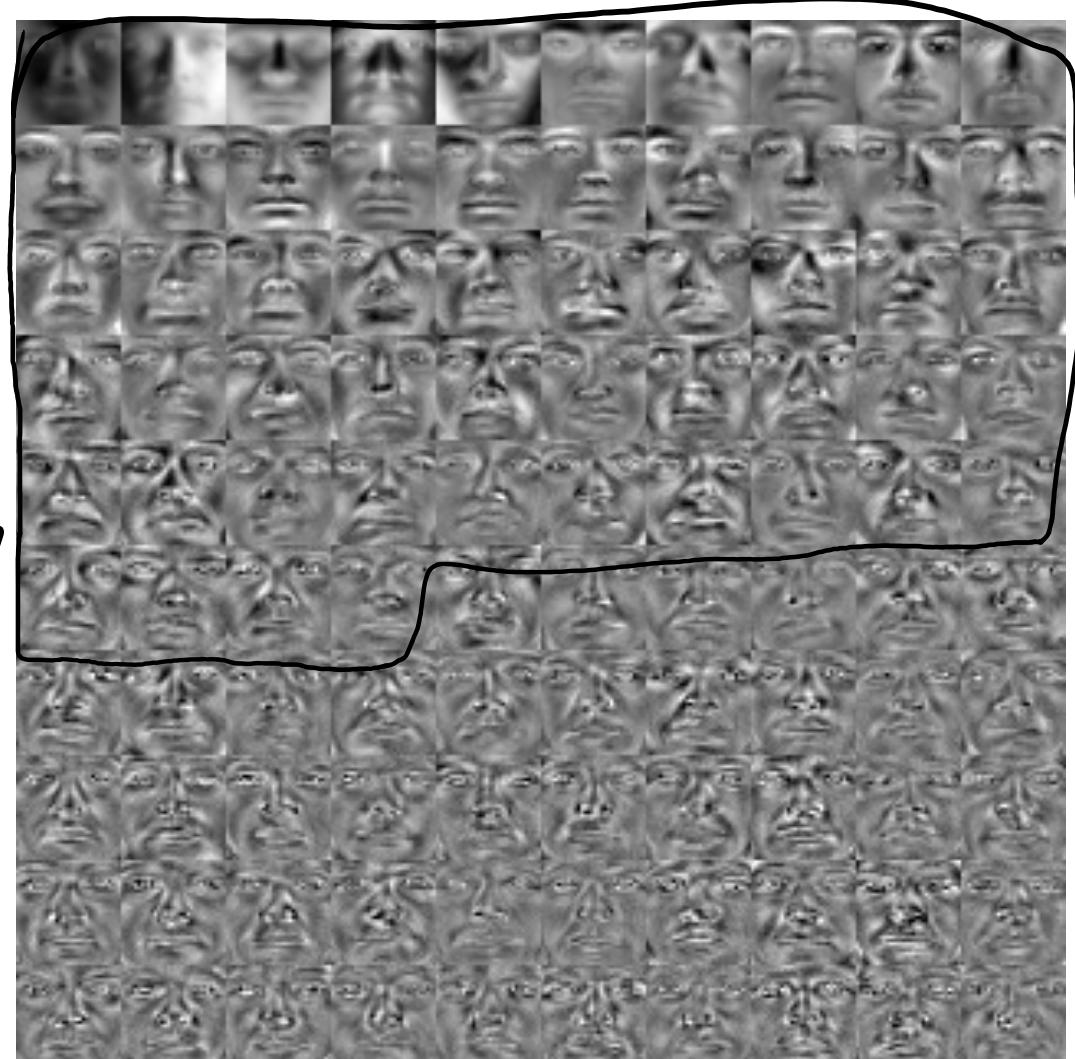


# Eigenfaces

Original Images again:



Plus these  
"eigenfaces"  
and  
the  
mean,



We can replace 1024  $x_i$  values by 54  $z_i$  values

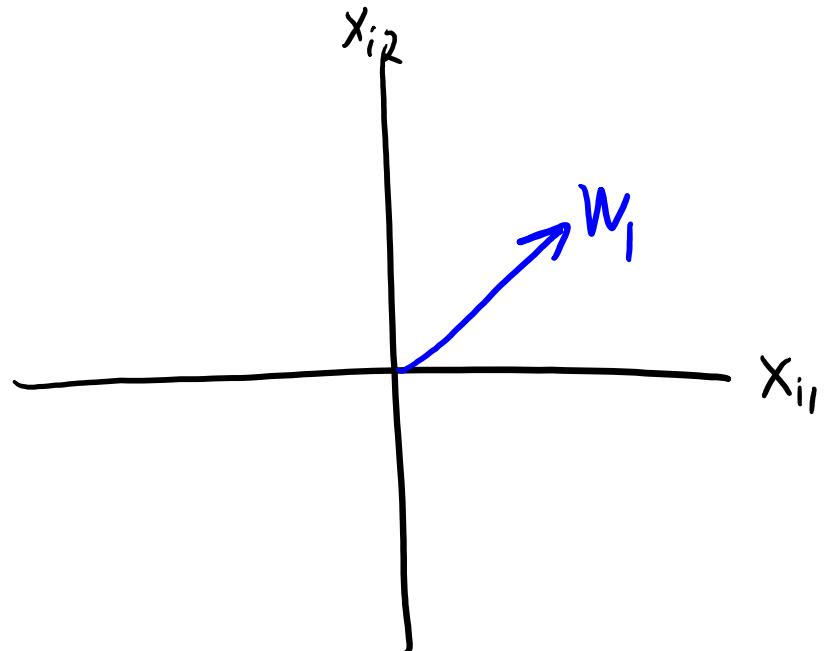
(pause)

# Non-Uniqueness of PCA

- Unlike k-means, we **can efficiently find global optima** of  $f(W, Z)$ .
  - Algorithms coming later.
- Unfortunately, there **never exists a unique global optimum**.
  - There are actually several different sources of non-uniqueness.
- To understand these, we'll need idea of “**span**” from linear algebra.
  - This also helps explain the geometry of PCA.
  - We'll also see that **some global optima may be better than others**.

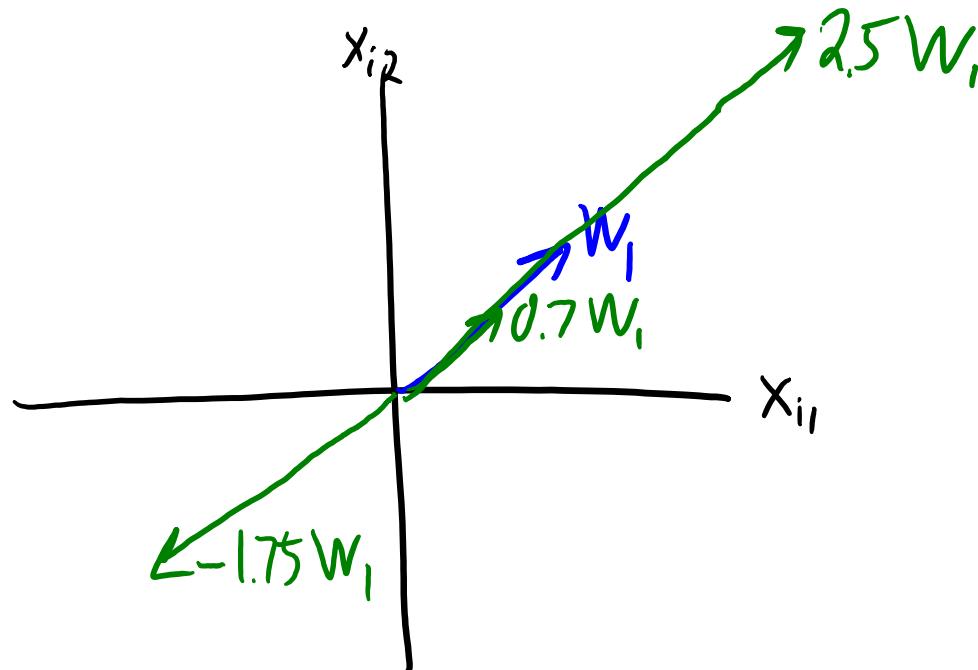
# Span of 1 Vector

- Consider a single vector  $w_1$  ( $k=1$ ).



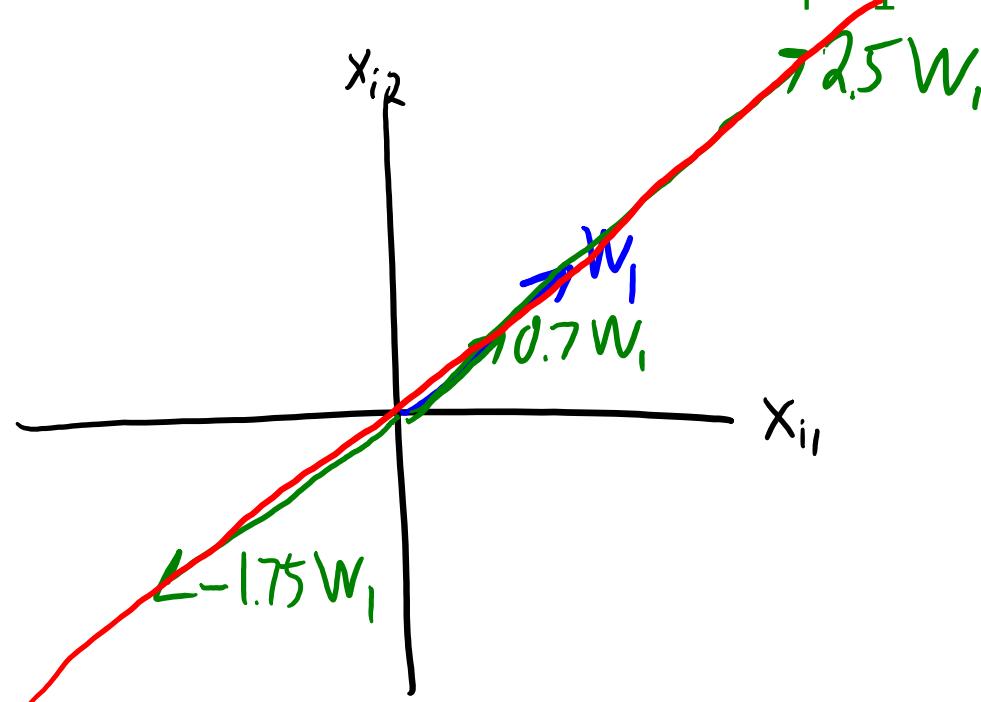
# Span of 1 Vector

- Consider a single vector  $w_1$  ( $k=1$ ).
- The  $\text{span}(w_1)$  is all vectors of the form  $z_i w_1$  for a scalar  $z_i$ .



# Span of 1 Vector

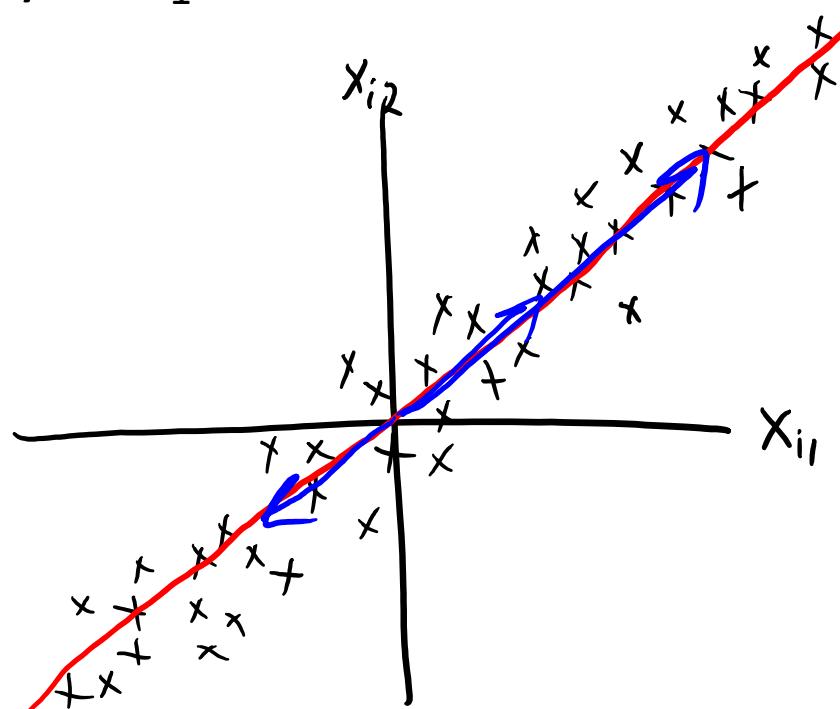
- Consider a single vector  $w_1$  ( $k=1$ ).
- The  $\text{span}(w_1)$  is all vectors of the form  $z_i w_1$  for a scalar  $z_i$ .



- If  $w_1 \neq 0$ , this forms a line.

# Span of 1 Vector

- But note that the “span” of many different vectors gives same line.
  - Mathematically:  $\alpha w_1$  defines the same line as  $w_1$  for any scalar  $\alpha \neq 0$ .

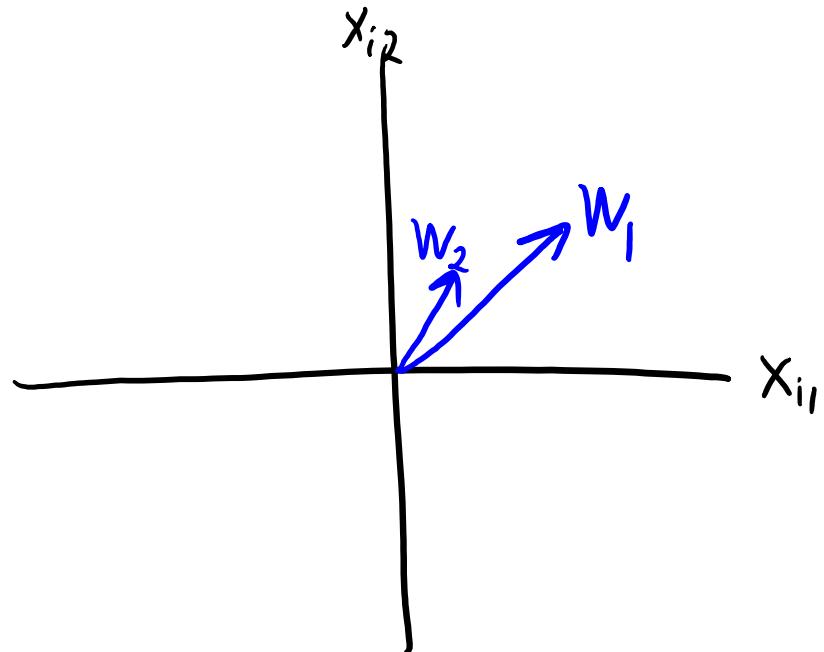


– PCA solution can only be defined up to scalar multiplication.

- If  $(W, Z)$  is a solution, then  $(\alpha W, (1/\alpha)Z)$  is also a solution.  $\|(\alpha W)(\frac{1}{\alpha}Z) - X\|_F^2 = \|WZ - X\|_F^2$

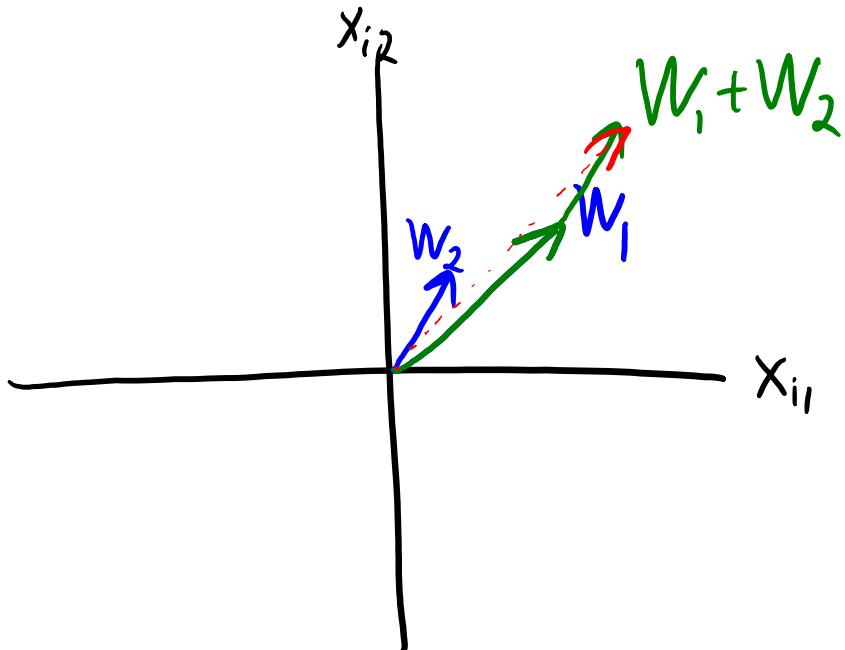
# Span of 2 Vectors

- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).



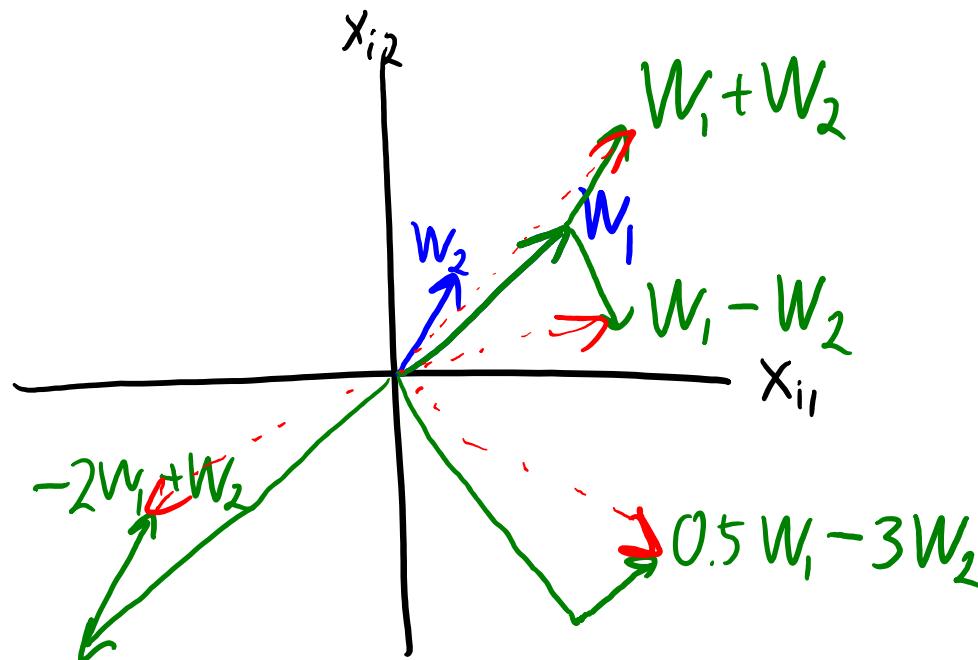
# Span of 2 Vectors

- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).
  - The  $\text{span}(w_1, w_2)$  is all vectors of form  $z_{i1}w_1 + z_{i2}w_2$  for a scalars  $z_{i1}$  and  $z_{i2}$ .



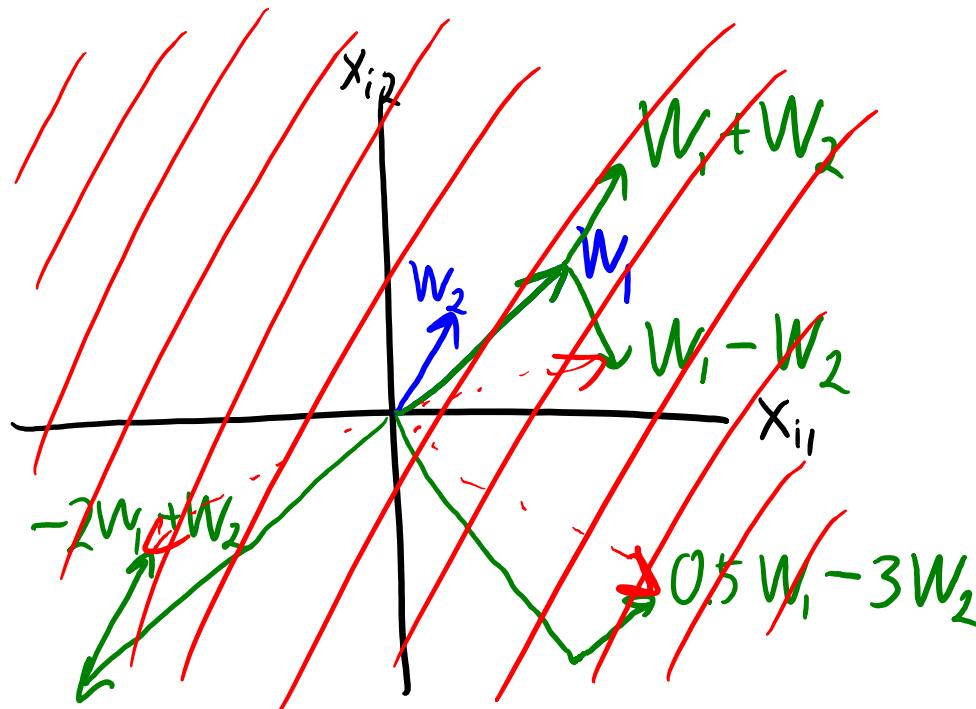
# Span of 2 Vectors

- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).
  - The  $\text{span}(w_1, w_2)$  is all vectors of form  $z_{i1}w_1 + z_{i2}w_2$  for a scalars  $z_{i1}$  and  $z_{i2}$ .



# Span of 2 Vectors

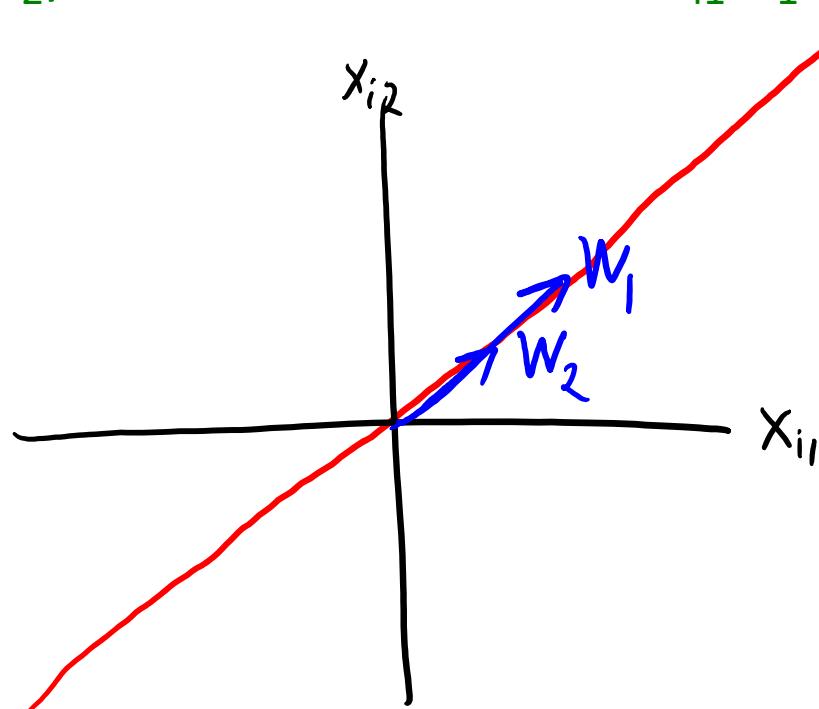
- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).
  - The  $\text{span}(w_1, w_2)$  is all vectors of form  $z_{i1}w_1 + z_{i2}w_2$  for a scalars  $z_{i1}$  and  $z_{i2}$ .



- For most non-zero 2d vectors,  $\text{span}(w_1, w_2)$  is a plane.
  - In the case of two vectors in  $R^2$ , the plane will be \*all\* of  $R^2$ .

# Span of 2 Vectors

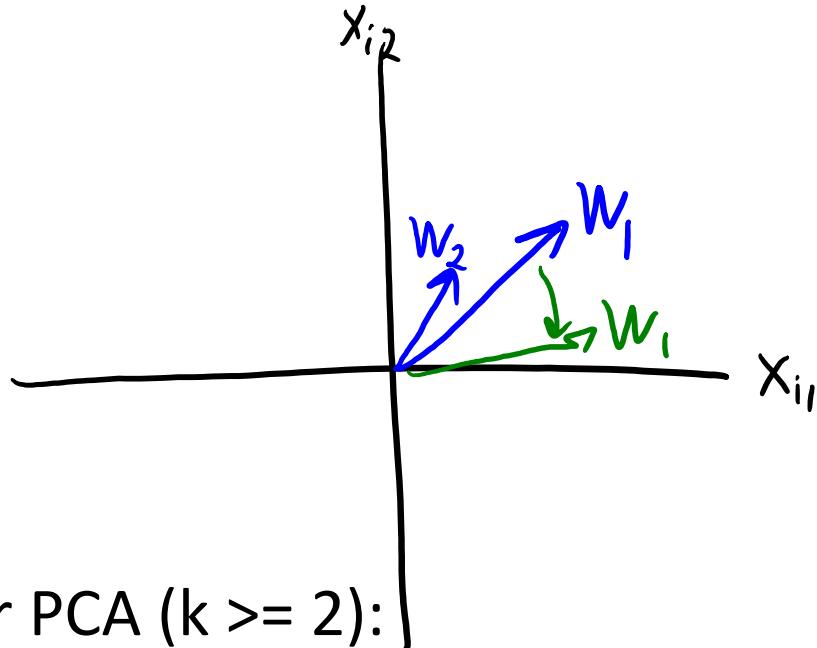
- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).
  - The  $\text{span}(w_1, w_2)$  is all vectors of form  $z_{i1}w_1 + z_{i2}w_2$  for a scalars  $z_{i1}$  and  $z_{i2}$ .



- For most non-zero 2d vectors,  $\text{span}(w_1, w_2)$  is plane.
  - Exception is if  $w_2$  is in  $\text{span}$  of  $w_1$  ("collinear"), then  $\text{span}(w_1, w_2)$  is just a line.

# Span of 2 Vectors

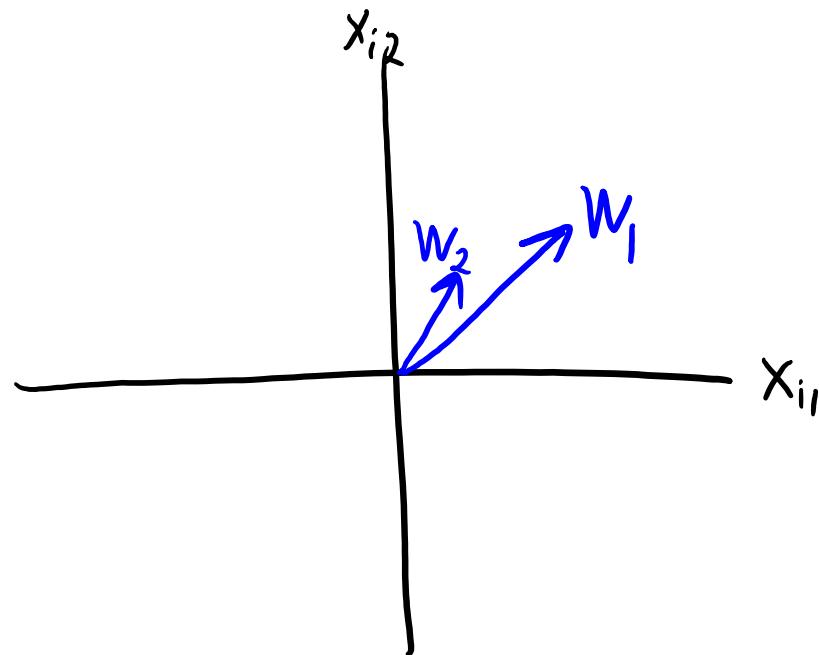
- Consider two vector  $w_1$  and  $w_2$  ( $k=2$ ).
  - The  $\text{span}(w_1, w_2)$  is all vectors of form  $z_{i1}w_1 + z_{i2}w_2$  for a scalars  $z_{i1}$  and  $z_{i2}$ .



- New issues for PCA ( $k \geq 2$ ):
  - We have **label switching**:  $\text{span}(w_1, w_2) = \text{span}(w_2, w_1)$ .
  - We can **rotate factors** within the plane (if not rotated to be collinear).

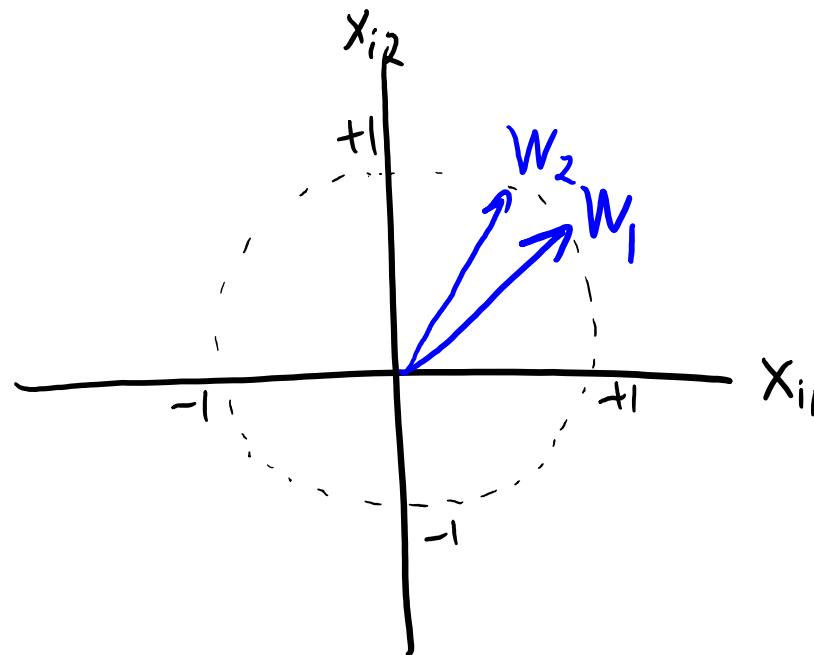
# Span of 2 Vectors

- 2 tricks to make vectors defining a plane “more unique”:
  - Normalization: enforce that  $\|w_1\| = 1$  and  $\|w_2\| = 1$ .



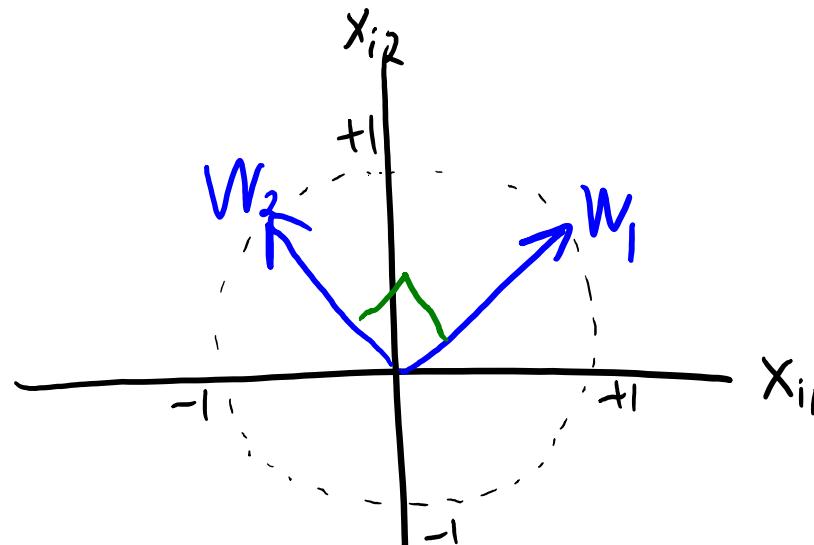
# Span of 2 Vectors

- 2 tricks to make vectors defining a plane “more unique”:
  - **Normalization**: enforce that  $\|w_1\| = 1$  and  $\|w_2\| = 1$ .



# Span of 2 Vectors

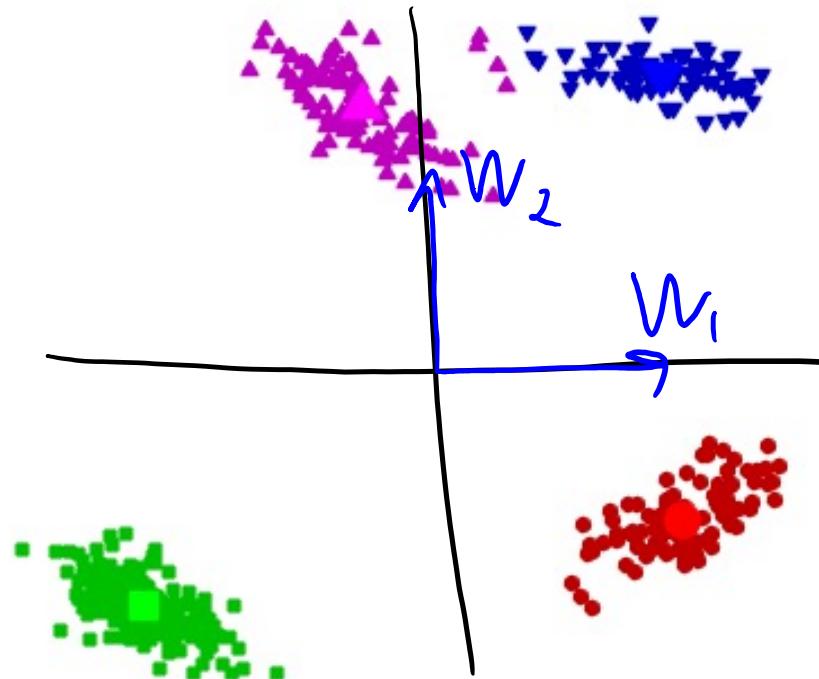
- 2 tricks to make vectors defining a plane “more unique”:
  - Normalization: enforce that  $\|w_1\| = 1$  and  $\|w_2\| = 1$ .
  - Orthogonality: enforce that  $w_1^T w_2 = 0$  (“perpendicular”).



- Now I can’t grow/shrink vectors (though I can still reflect).
- Now I can’t rotate one vector (but I can still rotate \*both\*).

# Digression: PCA only makes sense for $k \leq d$

- Remember our clustering dataset with 4 clusters:



- It **doesn't make sense to use PCA with  $k=4$**  on this dataset.
  - We **only need two vectors**  $[1 \ 0]$  and  $[0 \ 1]$  to exactly represent all 2d points.
    - With  $k=2$ , I could set  $Z=X$  and  $W=I$  to get  $X=ZW$  exactly.

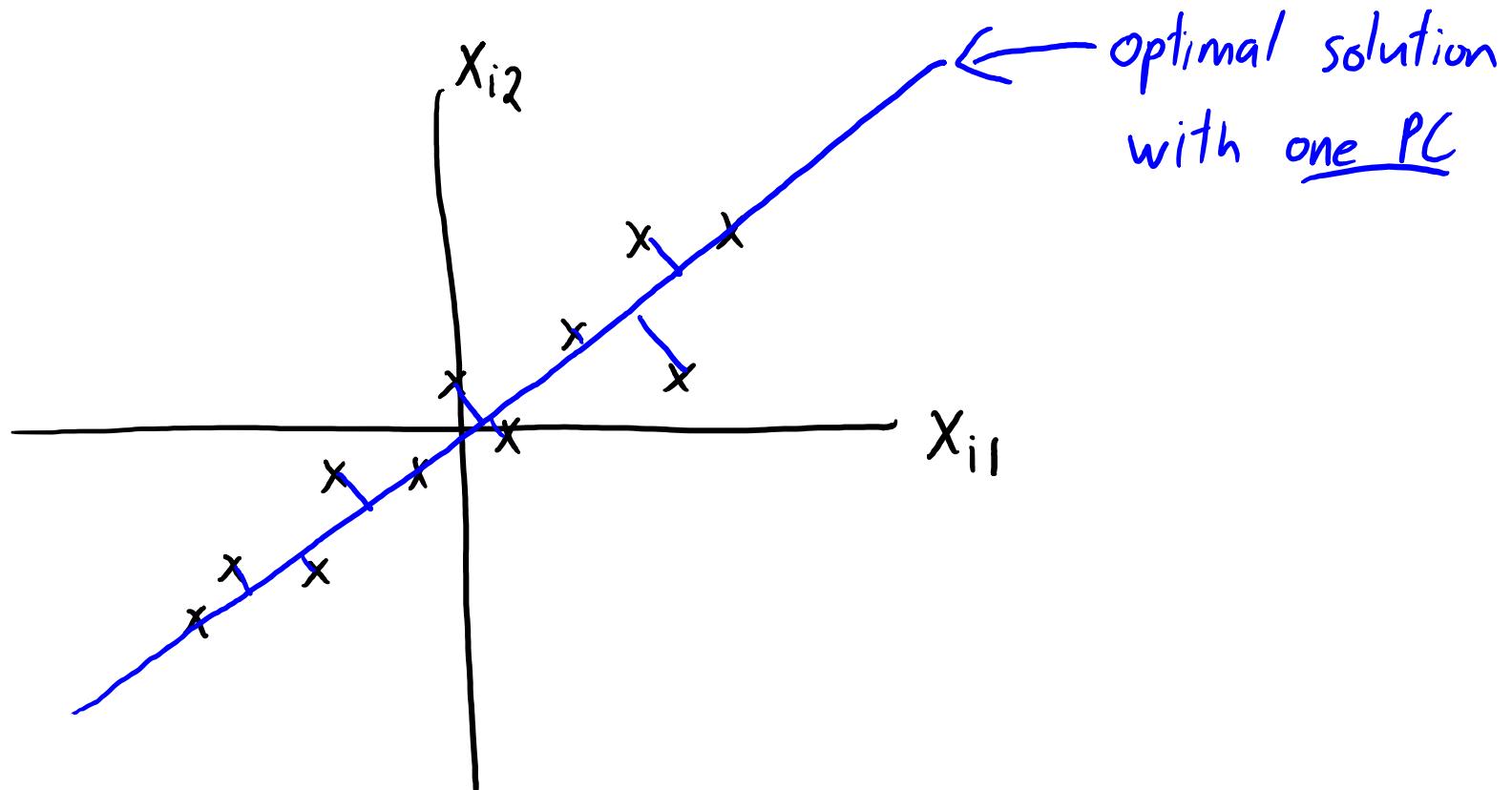
# Span in Higher Dimensions

- In higher-dimensional spaces:
  - Span of 1 non-zero vector  $w_1$  is a line.
  - Span of 2 non-zero vectors  $w_1$  and  $w_2$  is a plane (if not collinear).
    - Can be visualized as a 2D plot.
  - Span of 3 non-zeros vectors  $\{w_1, w_2, w_3\}$  is a 3d space (if not “coplanar”).
  - ...
- This is how the  $W$  matrix in PCA defines lines, planes, spaces, etc.
  - Each time we increase ‘ $k$ ’, we add an extra “dimension” to the “subspace”.

# Making PCA Unique

- We've identified several reasons that optimal  $W$  is non-unique:
  - I can multiply any  $w_c$  by any non-zero  $\alpha$ .
  - I can rotate any  $w_c$  almost arbitrarily within the span.
  - I can switch any  $w_c$  with any other  $w_{c'}$ .
- PCA implementations add constraints to make solution unique:
  - Normalization: we enforce that  $\|w_c\| = 1$ .
  - Orthogonality: we enforce that  $w_c^T w_{c'} = 0$  for all  $c \neq c'$ .
  - Sequential fitting: We first fit  $w_1$  ("first principal component") giving a line.
    - Then fit  $w_2$  given  $w_1$  ("second principal component") giving a plane.
    - Then we fit  $w_3$  given  $w_1$  and  $w_2$  ("third principal component") giving a space.

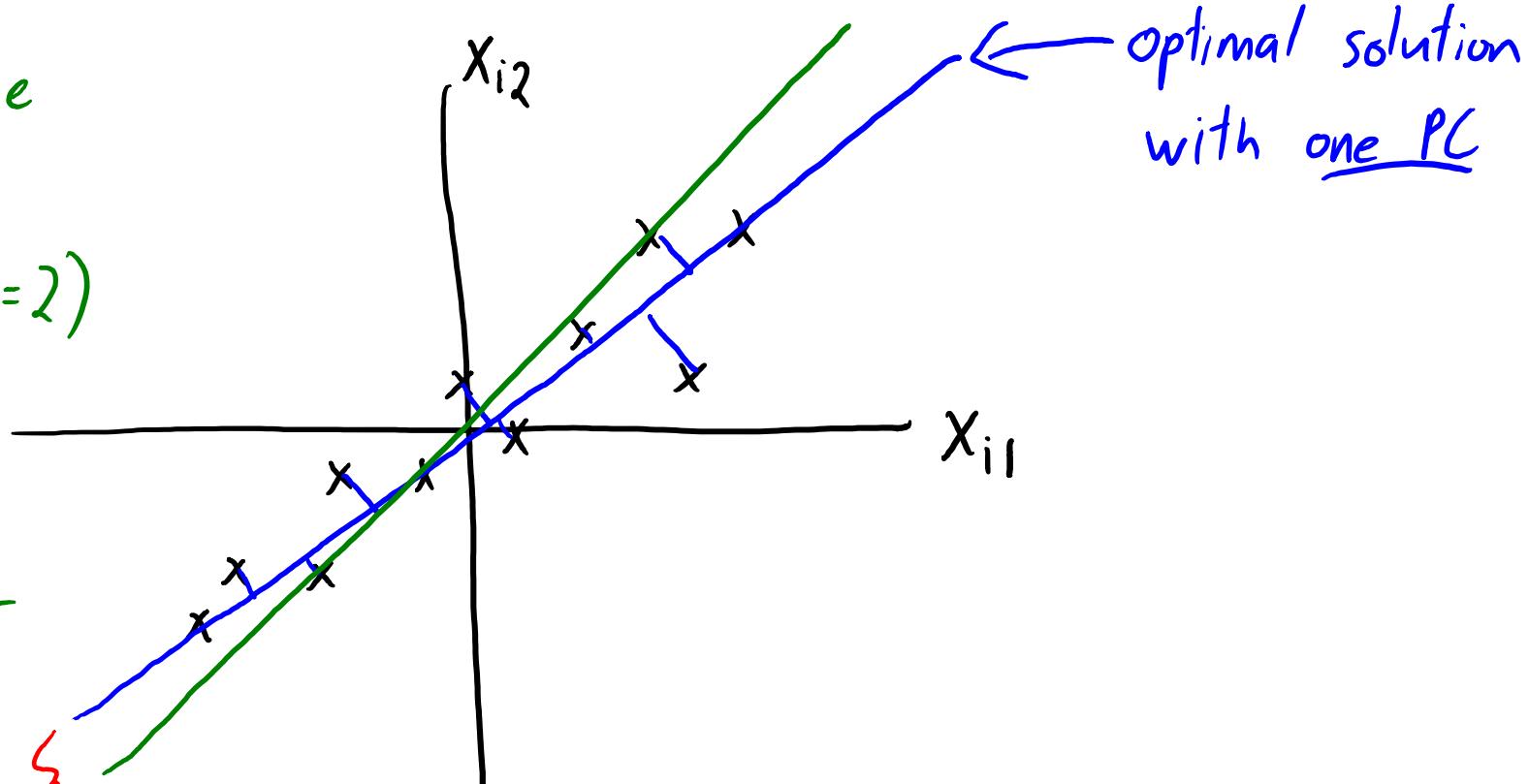
# Basis, Orthogonality, Sequential Fitting



# Basis, Orthogonality, Sequential Fitting

Any non-parallel line  
gives optimal solution  
to second PC (when  $d=2$ )

I can get 0 error  
on every data point.



An optimal solution but not orthogonal.

(both PCs give similar information)

# Basis, Orthogonality, Sequential Fitting

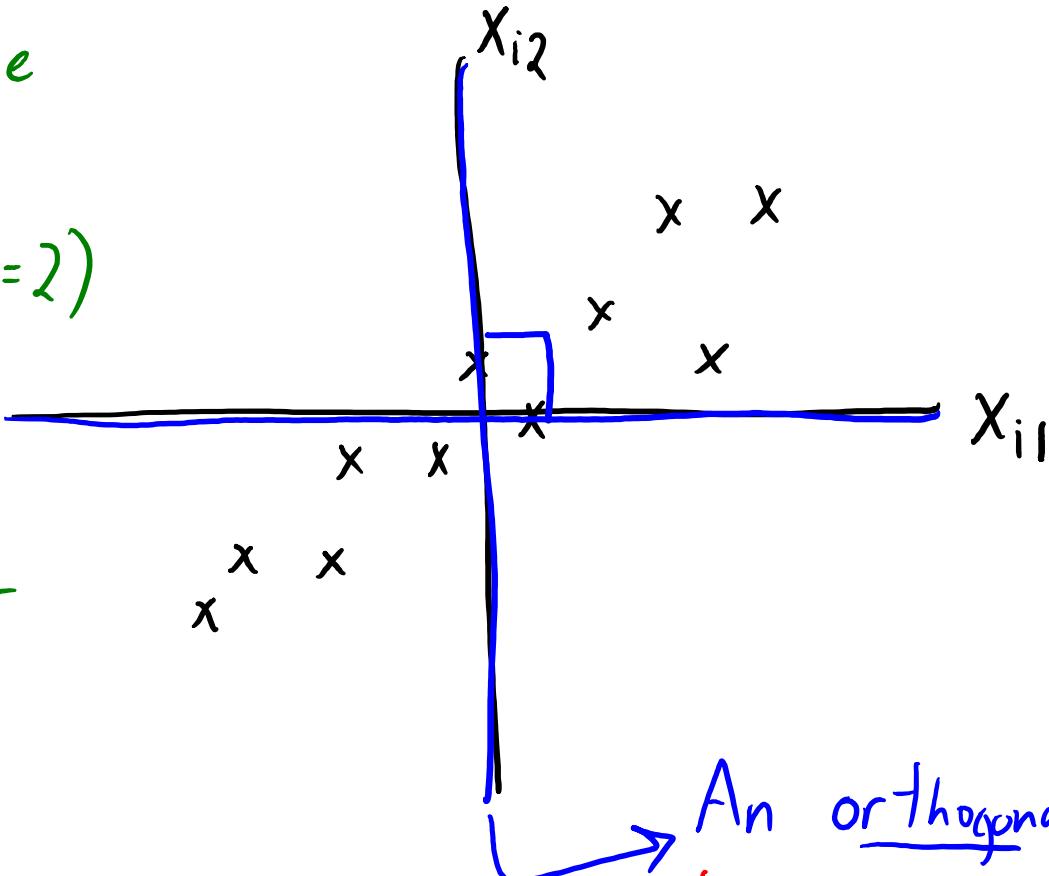
Any non-parallel line

gives optimal solution

to second PC (when  $d=2$ )

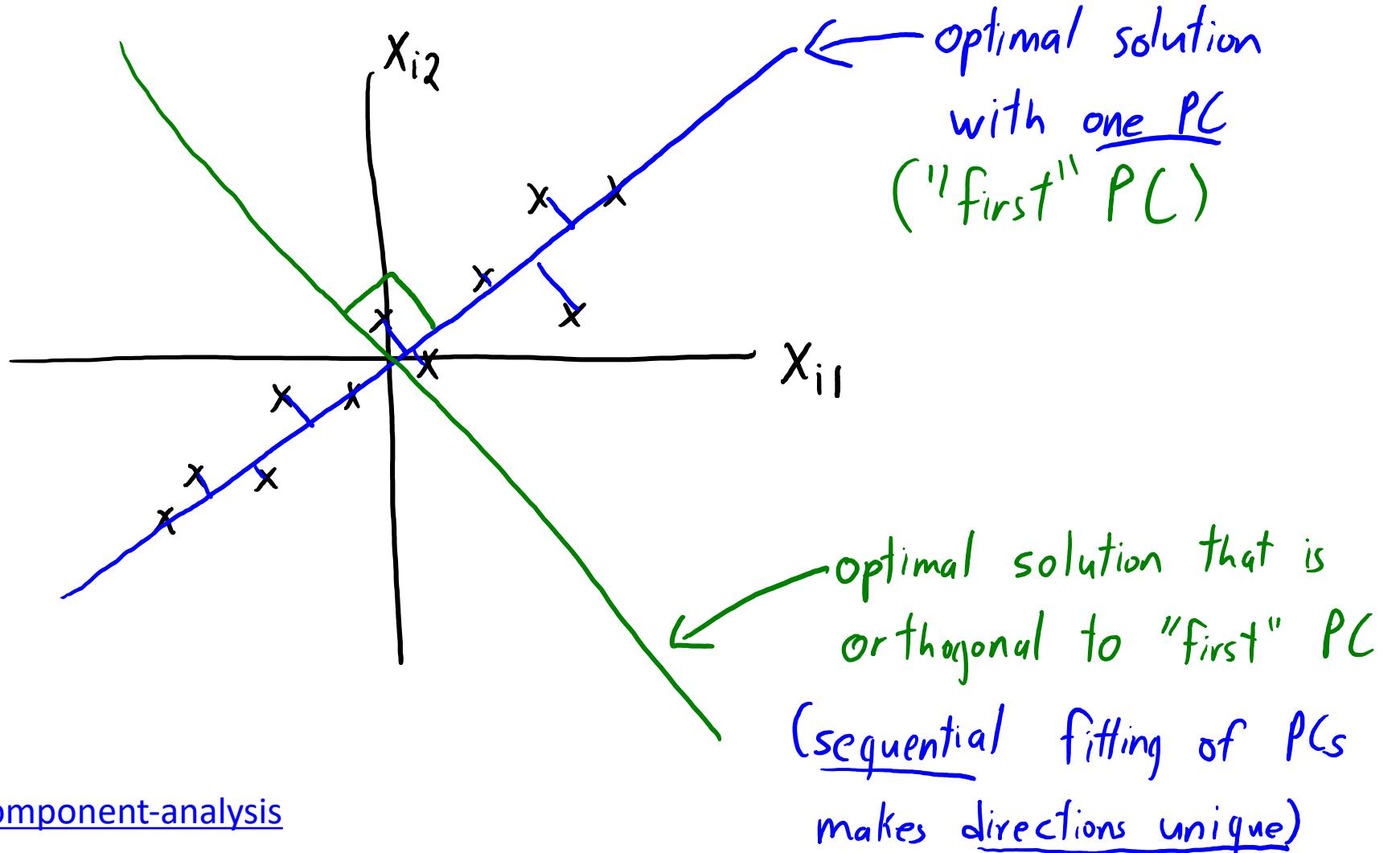


I can get 0 error  
on every data point.



An orthogonal solution (PCs are not redundant)  
but PCs have nothing to do with data

# Basis, Orthogonality, Sequential Fitting



bonus!

# PCA Computation: SVD

- How do we fit with normalization/orthogonality/sequential-fitting?
  - It can be done with the “singular value decomposition” (SVD).
  - Take CPSC 302.
- 4 lines of Python code:
  - `mu = np.mean(X, axis=0)`
  - `X -= mu`
  - `U, s, Vh = np.linalg.svd(X)`
  - `W = Vh[:k]`
- Computing Z is cheaper now:
$$Z = X W^\top (WW^\top)^{-1} = X W^\top$$
$$WW^\top = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix} \begin{bmatrix} 1 & 1 & \dots & 1 \\ w_1^\top & w_2^\top & \dots & w_k^\top \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & 1 \end{bmatrix} = I$$

# Summary

- PCA objective:
  - Minimizes squared error between elements of  $X$  and elements of  $ZW$ .
- Choosing ‘k’:
  - We can choose ‘k’ to explain “percentage of variance” in the data.
- PCA non-uniqueness:
  - Due to scaling, rotation, and label switching.
- Orthogonal basis and sequential fitting of PCs (via SVD):
  - Leads to non-redundant PCs with unique directions.
- Next time: cancer signatures and NBA shot charts.

bonus!

# Making PCA Unique

- PCA implementations add **constraints** to make solution unique:
  - **Normalization**: we enforce that  $\|w_c\| = 1$ .
  - **Orthogonality**: we enforce that  $w_c^T w_{c'} = 0$  for all  $c \neq c'$ .
  - **Sequential fitting**: We first fit  $w_1$  (“first principal component”) giving a line.
    - Then fit  $w_2$  given  $w_1$  (“second principal component”) giving a plane.
    - Then we fit  $w_3$  given  $w_1$  and  $w_2$  (“third principal component”) giving a space.
    - ...
- Even with all this, the solution is **only unique up to sign changes**:
  - I can still replace any  $w_c$  by  $-w_c$ :
    - $-w_c$  is normalized, is orthogonal to the other  $w_{c'}$ , and spans the same space.
  - Possible fix: require that first non-zero element of each  $w_c$  is positive.
  - And this is assuming you don’t have repeated singular values.
    - In that case you can rotate the repeated ones within the same plane.