

CPSC 340: Machine Learning and Data Mining

More PCA

Admin

- My office hours:
 - I've had to reschedule them from today to tomorrow 2-4pm
 - Sorry for the short notice
- Assignment 4:
 - Due Sunday.
- Assignment 5:
 - Coming soon.

The 10 Algorithms Machine Learning Engineers Need to Know

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



Last time: Principal Component Analysis (PCA)

- PCA is a linear model for unsupervised learning.
 - Represents features as linear combination of latent factors:

$$x_{ij} = w_j^T z_i$$

↳ column j

$$x_i = W^T z_i = z_{i1} w_1 + z_{i2} w_2 + \dots + z_{ik} w_k$$

W' and latent features z_i .

late matrix factorization:

- But we’re learning the latent factors ‘W’ and latent features z_i .

- Can also be viewed as an approximate matrix factorization:

$$X \approx ZW$$

$$\left[\begin{array}{c} n \times d \\ \downarrow \end{array} \right] \approx \left[\begin{array}{c} n \times K \\ \vdots \end{array} \right] \left[\begin{array}{c} K \times d \end{array} \right]$$

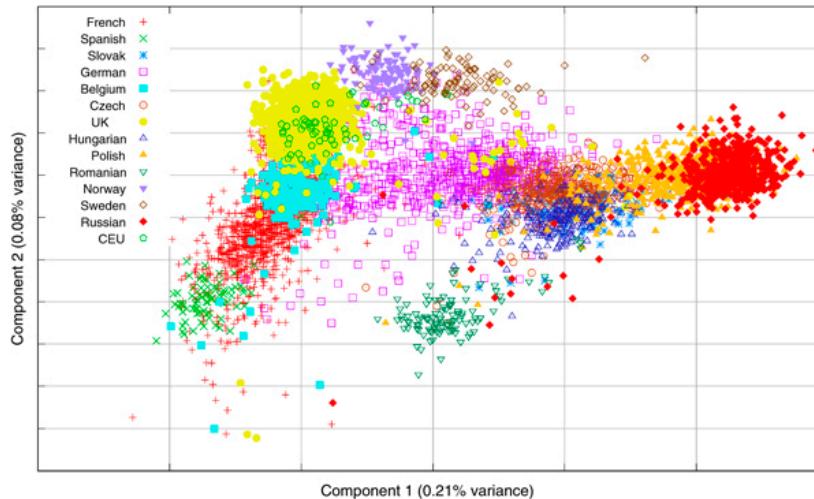
Last time: Principal Component Analysis (PCA)

- PCA is a linear model for unsupervised learning.
- Represents features as linear combination of latent factors:

$$x_{ij} = w_j^T z_i$$

$$x_i = W^T z_i$$

- 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.

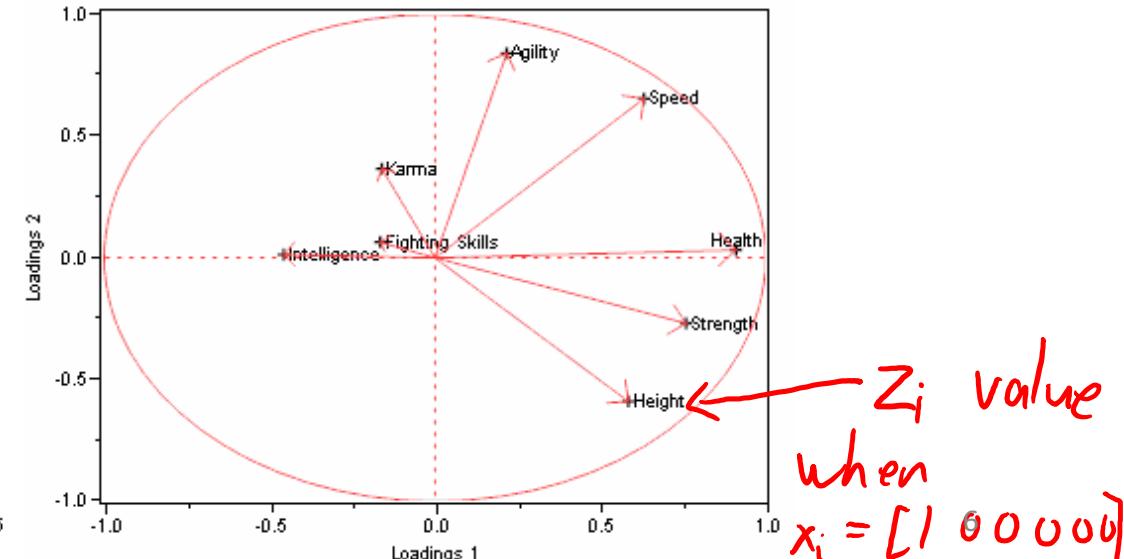
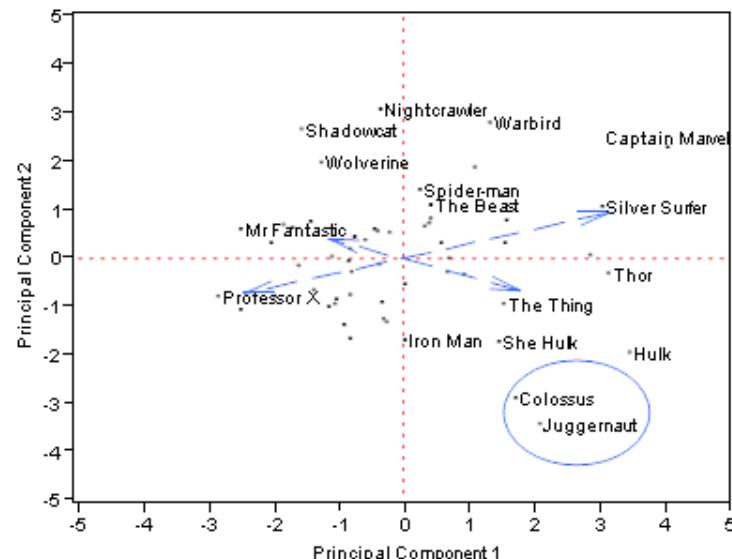
Last time: Principal Component Analysis (PCA)

- PCA is a linear model for unsupervised learning.
- Represents features as linear combination of latent factors:

$$x_{ij} = w_j^T z_i$$

$$x_i = W^T z_i$$

- Uses: dimensionality reduction, visualization, factor discovery.



PCA with d=2 and k =1

- So simplest interesting case is d=2 and k=1:

$$X = \begin{bmatrix} \quad \end{bmatrix} \quad n \times 2$$

$$Z = \begin{bmatrix} \quad \end{bmatrix} \quad n \times 1$$

$$W = \begin{bmatrix} \quad \\ 1 \times 2 \end{bmatrix}$$

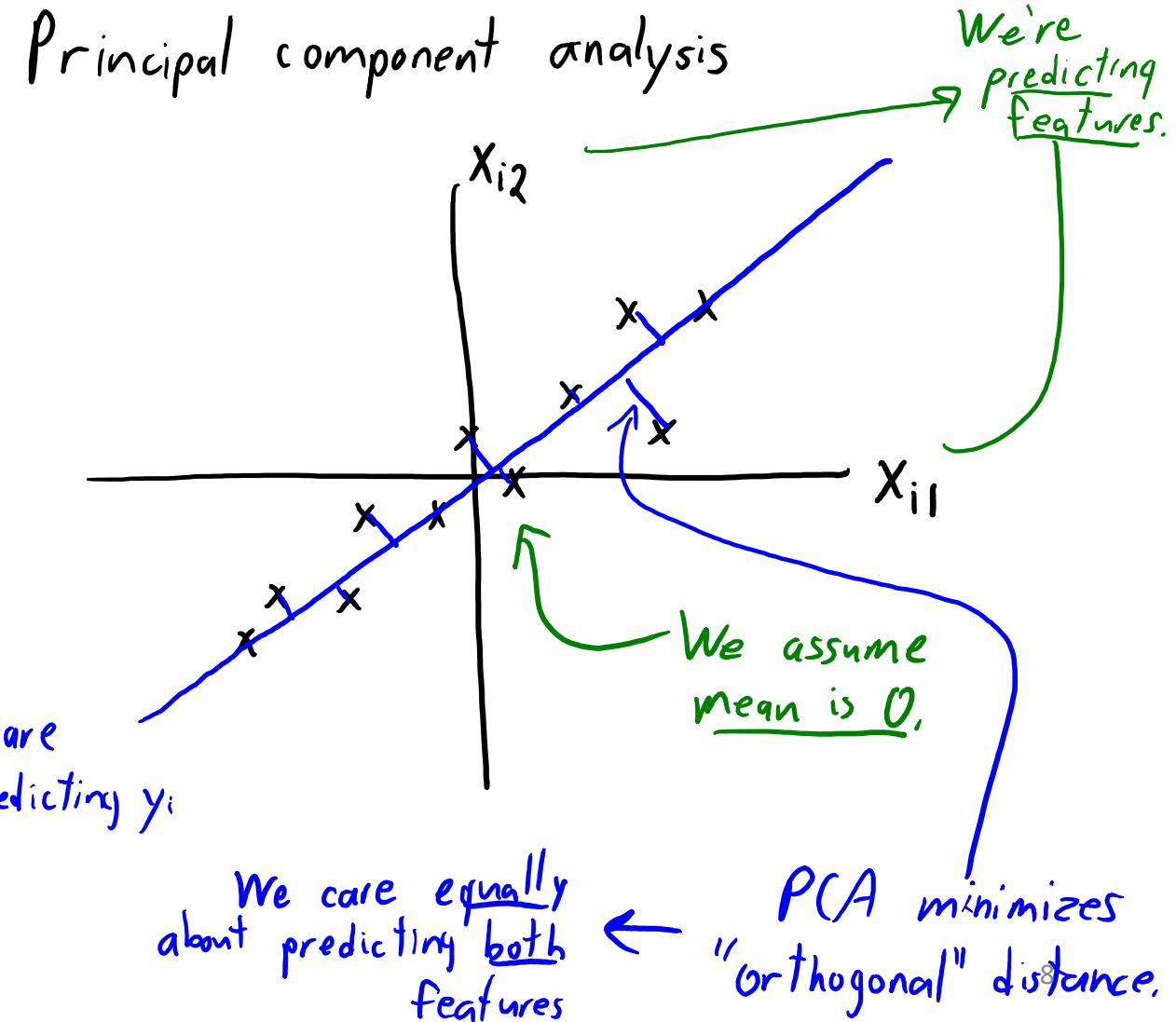
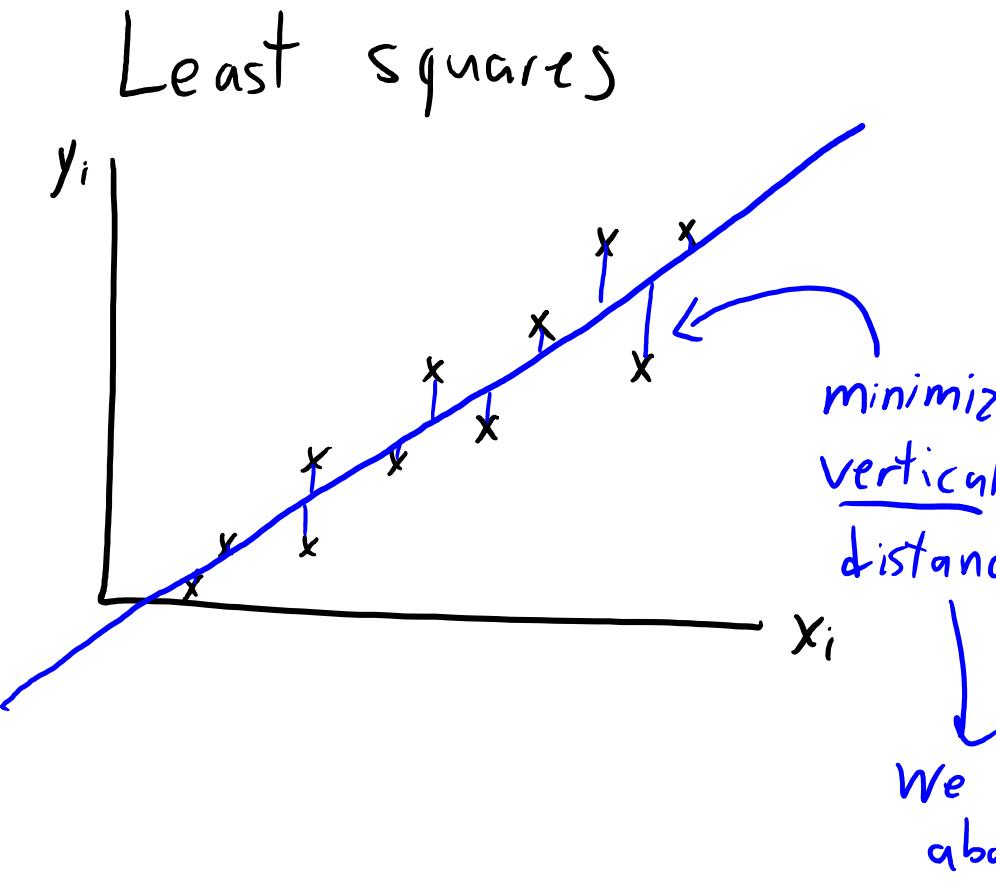
New "feature"
for example 'i'
"Principal component"

PCA objective:

$$\begin{aligned} f(Z, W) &= \sum_{i=1}^n \sum_{j=1}^2 (w_j z_i - x_{ij})^2 \\ &= \sum_{i=1}^n (w_1 z_i - x_{i1})^2 + \sum_{i=1}^n (w_2 z_i - x_{i2})^2 \end{aligned}$$

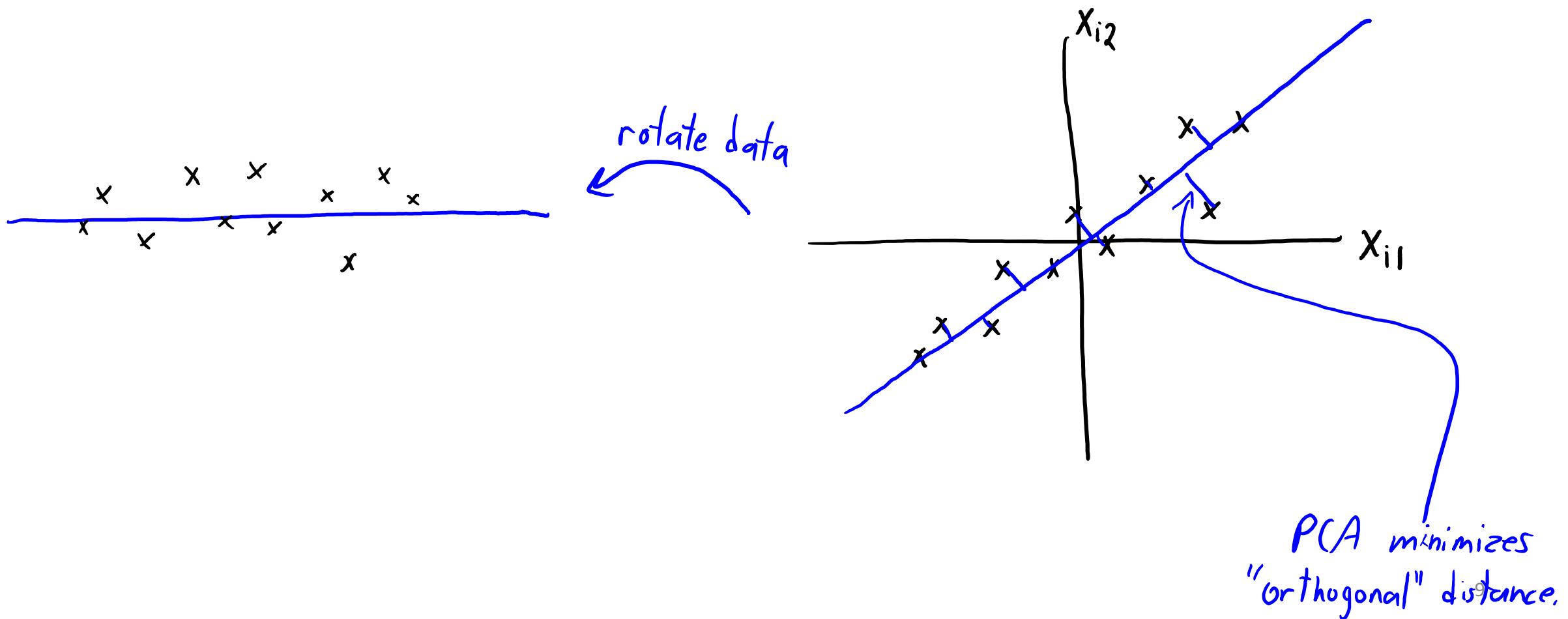
- Very similar to a least squares problem, but note that:
 - We have no ' y_i ', we are trying to predict each feature x_{ij} from the single z_i .
 - But features ' z_i ' are also variables, we are learning the features z_i too.
- Side note: in PCA we assume features have a mean of 0.
 - You can subtract mean or add bias variable if this is not true.

PCA with $d=2$ and $k = 1$



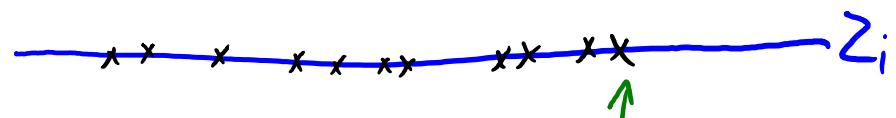
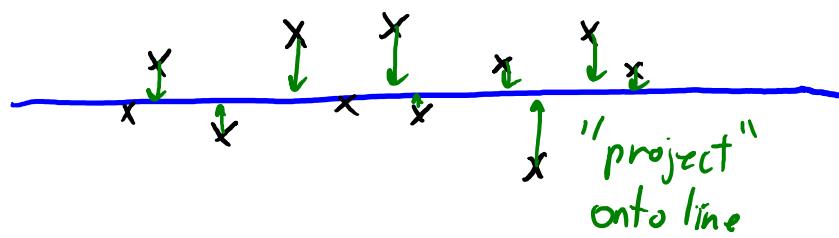
PCA with $d=2$ and $k = 1$

Principal component analysis

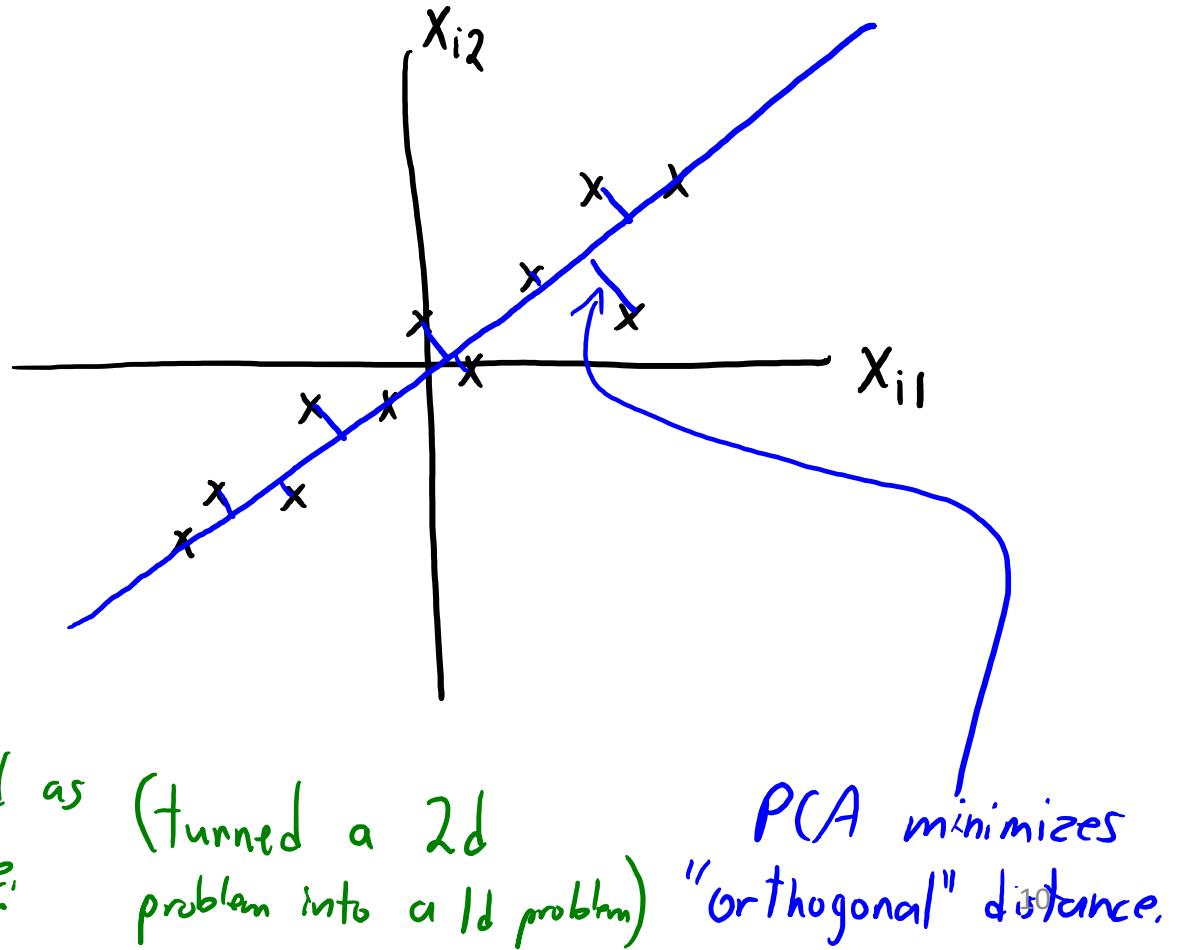


PCA with $d=2$ and $k = 1$

Principal component analysis



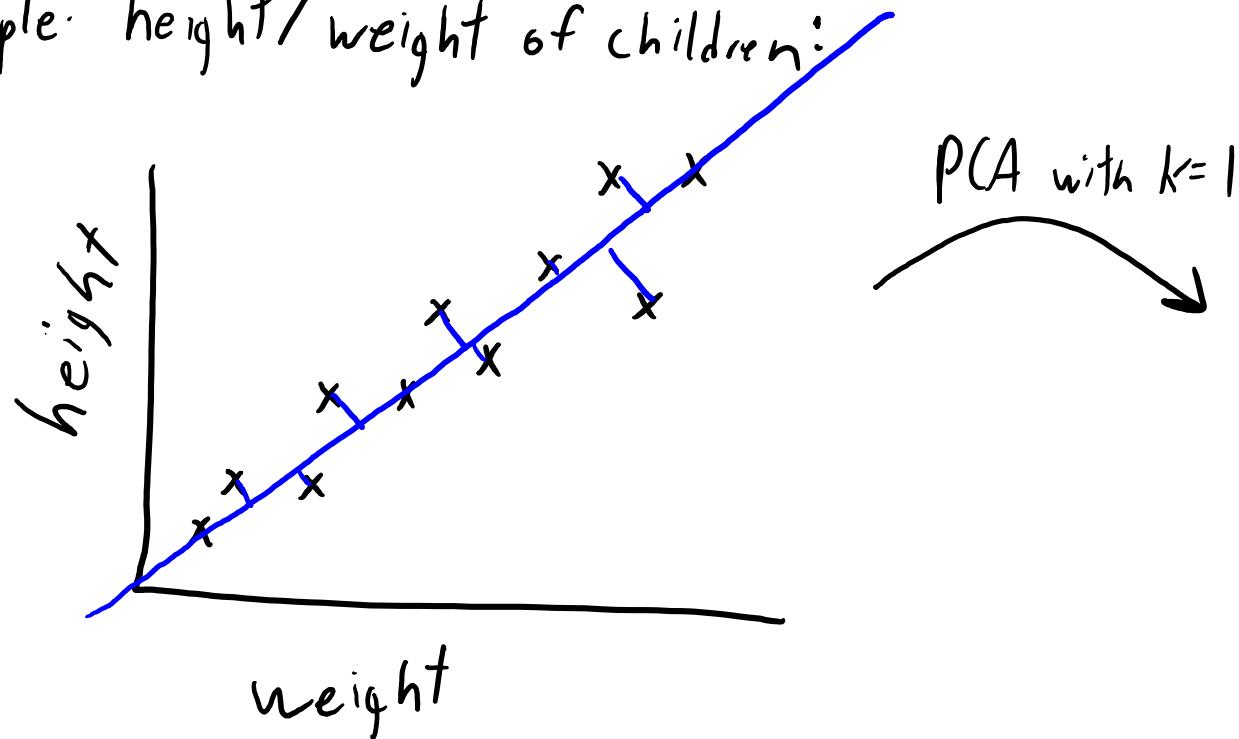
Z_i can be interpreted as
position along the line



PCA minimizes
"orthogonal" distance.
10

PCA with $d=2$ and $k=1$

Example: height/weight of children:



Latent factor could be viewed as measure of size.

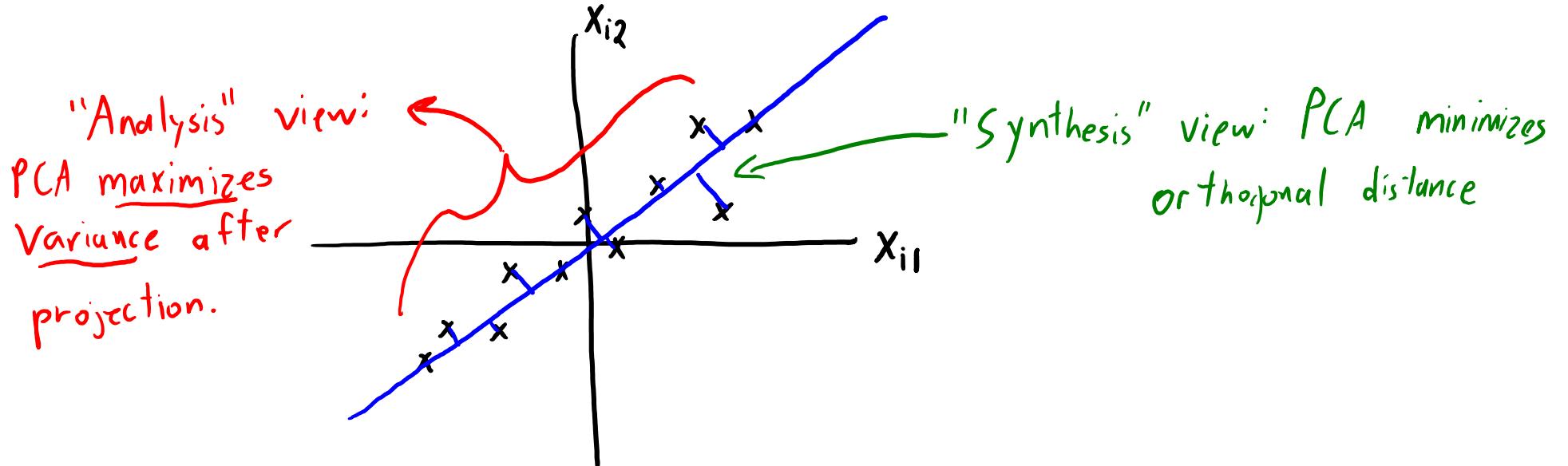
- The PCA objective with general 'd' and 'k':

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d (w_j^\top z_i - x_{ij})^2 = \|ZW - X\|_F^2$$

- Where we've subtracted mean μ_j from each feature.

Maximizing Variance vs. Minimizing Error

- Our “synthesis” view that PCA minimizes approximation error.
 - Makes connection to k-means and our tricks for linear regression.



- Classic “analysis” view: PCA maximizes variance in z_i space.
 - You pick ‘W’ to explain as much variance in the data as possible.

Choosing Number of Latent Factors

- Common approach to choosing ‘k’:

- Compute error with $k=0$: $\| X \|_F^2 = n * \text{var}(x_{ij})$

(remember that
columns have
mean of zero)

- Compare to error with non-zero ‘k’:

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

- Gives a number between 0 and 1, giving how much “variance remains”.
 - If you want to explain 90% of variance, choose smallest ‘k’ where ratio is < 0.10.

PCA Computation

- 3 common ways to solve this problem:
 - Singular value decomposition: classic non-iterative approach (bonus slide).
 - Alternating minimization:
 1. Start with random initialization.
 2. Optimize ‘W’ with ‘Z’ fixed (solve gradient with respect to ‘W’ equals to 0).

(writing gradient as a matrix) $\nabla_w f(w, z) = Z^T Z w - Z^T X \quad \text{so} \quad W = (Z^T Z)^{-1} (Z^T X)$

3. Optimize ‘Z’ with ‘W’ fixed (solve gradient with respect to ‘Z’ equals to 0).

$$\nabla_z f(w, z) = Z w w^T - X w^T \quad \text{so} \quad Z = X w^T (w w^T)^{-1}$$

4. Go back to 2.

- Stochastic gradient: gradient descent based on random ‘i’ and ‘j’.
 - (Or just plain gradient descent).

PCA Computation

- The PCA objective with general 'd' and 'k':

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d (w_j^\top z_i - x_{ij})^2 = \|Zw - X\|_F^2$$

- Where we've subtracted mean μ_j from each feature.
- At test time, to find optimal 'Z' given 'W' for new data:

Given factors 'W' and test data \hat{X} :

Subtract training mean μ_j for each feature 'j': $\hat{x}_i \leftarrow \hat{x}_i - \mu$

Solve for 'Z' given 'W': $Z = \hat{X} W^\top (WW^\top)^{-1}$

(If $k=1$ then $z_i = \frac{w_c^\top x_i}{w_c w_c}$)

PCA Non-Convexity

- The PCA objective with general ‘d’ and ‘k’:

$$f(W, Z) = \sum_{i=1}^n \sum_{j=1}^d (w_j^\top z_i - x_{ij})^2$$

- This objective is **not jointly convex** in ‘W’ and ‘Z’.
 - This is why **iterative methods need random initialization**.
 - If you initialize with $z_1 = z_2$, then they stay the same.
 - But it’s possible to show that **all “stable” local optima are global optima**.
 - So alternating minimization and stochastic gradient give **global optima** in practice.

PCA Non-Uniqueness

- Last time we discussed the **scaling** problem:

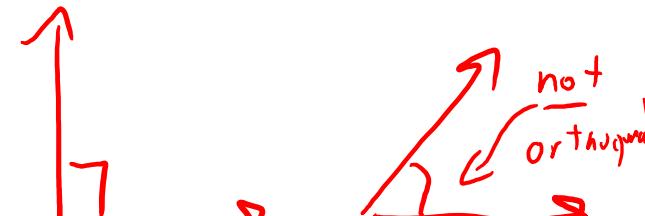
We get same $f(W, Z)$ if you replace ' W ' by αW
and ' Z ' by $(\frac{1}{\alpha})Z$ for any $\alpha \neq 0$

$$(\frac{1}{\alpha}Z)(\alpha W) = ZW$$

A standard fix: require that $\|w_c\| = 1$ for all factors ' c '.

↳ row ' c ' of ' W '

Orthogonality when $d=2$



- But with multiple PCs, we have new problems:

- Factors could be non-orthogonal (components interfere with each other):

For $d=2$ and $k=2$

an optimal solution is $W = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 1 \end{bmatrix}$

factors are almost identical

→ But with $d=2$ and $k=2$
we could equivalently
take $W = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Very interpretable

The standard fix is requiring orthogonal factors: $W_c^T W_{c'} = 0$ when $c \neq c'$

- You can still “rotate” the factors and also have label switching.

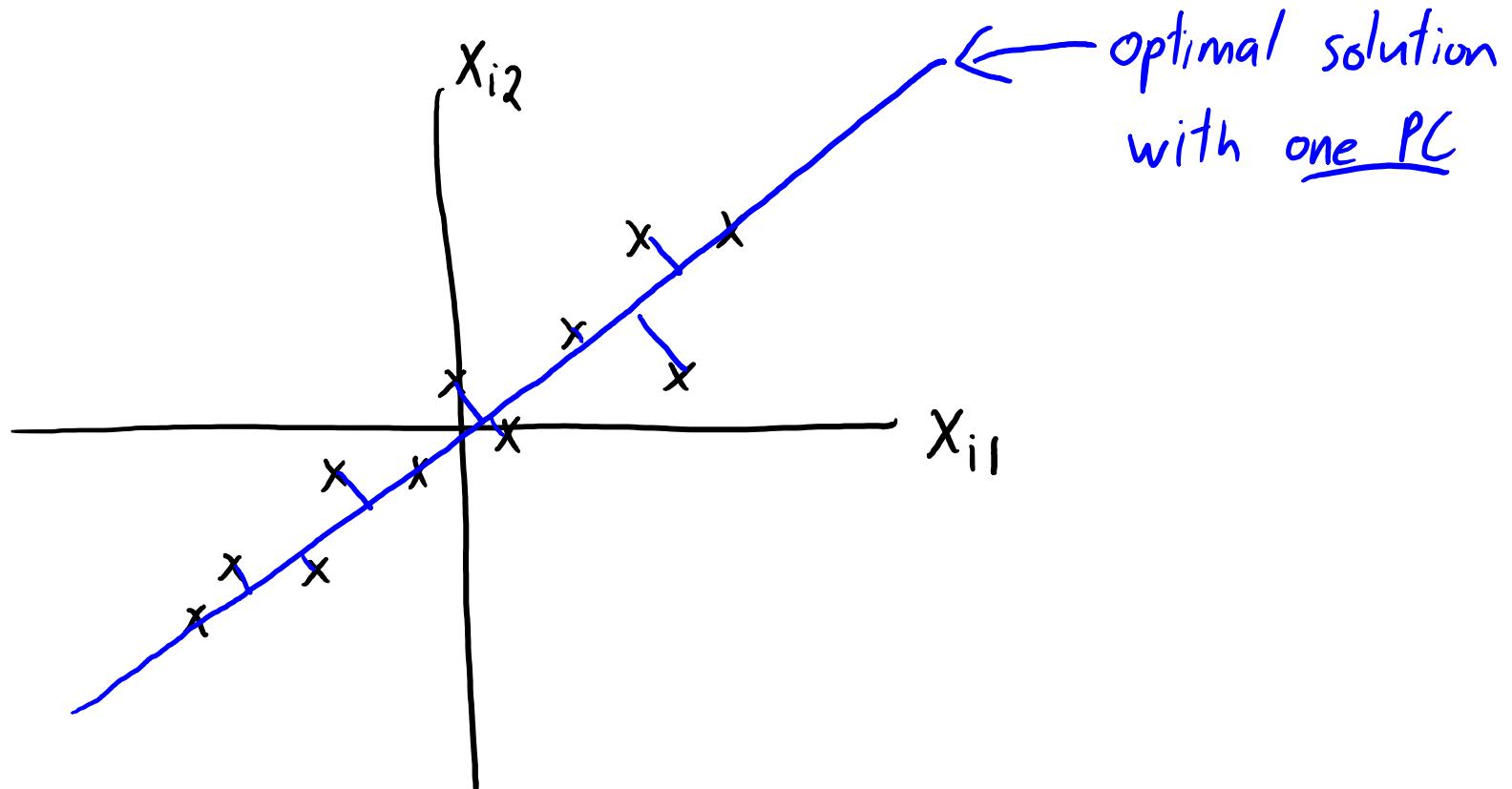
- A fix is to fit the PCs sequentially (can be done with SVD approach):

1. Find “first” PC w_c that minimizes $\|Zw_c^T - X\|_F^2$ (PCA with $k=1$)

2. Fix “first” PC w_1 and find w_c minimizing $\|Zw - X\|_F^2$ where $w_1^T w_c = 0$

3. Fix “first” and “second” PC and find w_c with $w_1^T w_c = 0$ and $w_2^T w_c = 0$

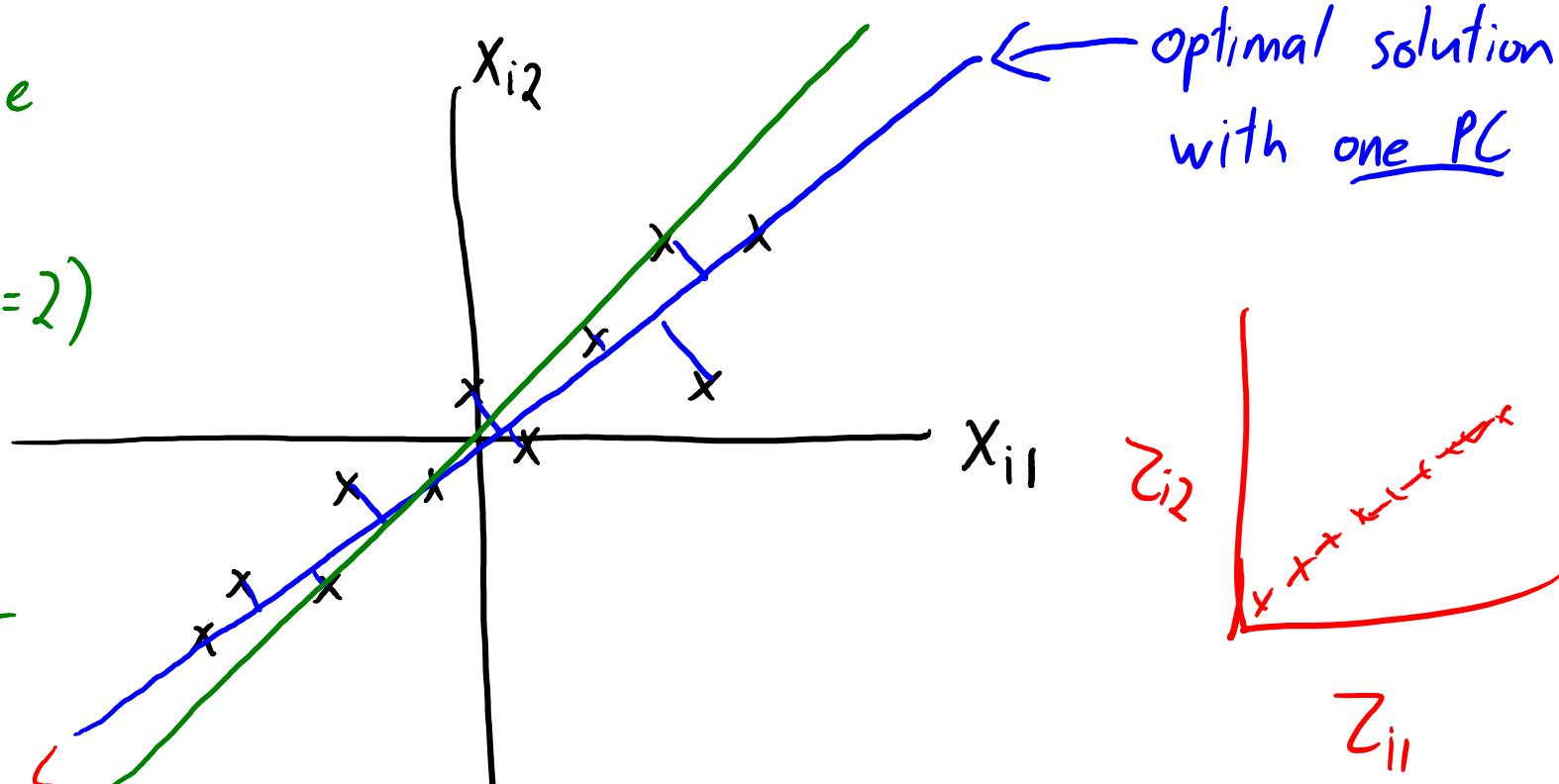
Basis, Orthogonality, Sequential Fitting



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.



(both PCs give similar information) ²⁰

Basis, Orthogonality, Sequential Fitting

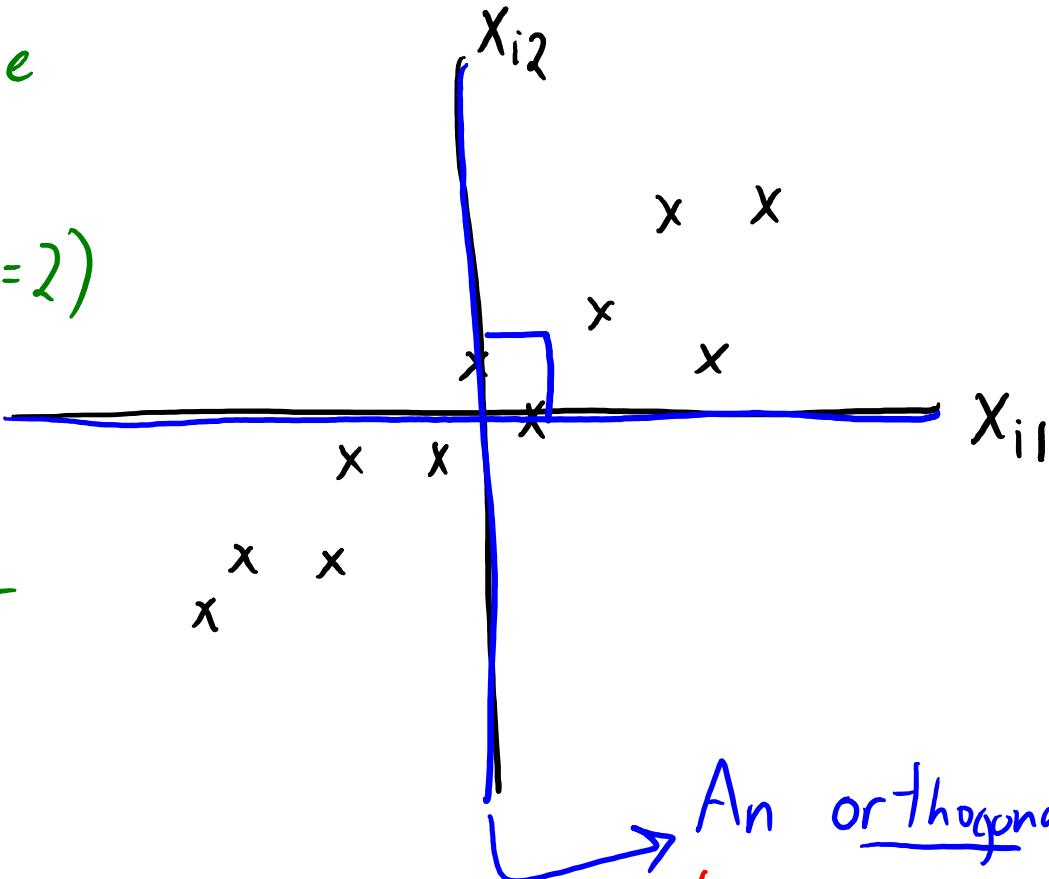
Any non-parallel line

gives optimal solution

to second PC (when $d=2$)

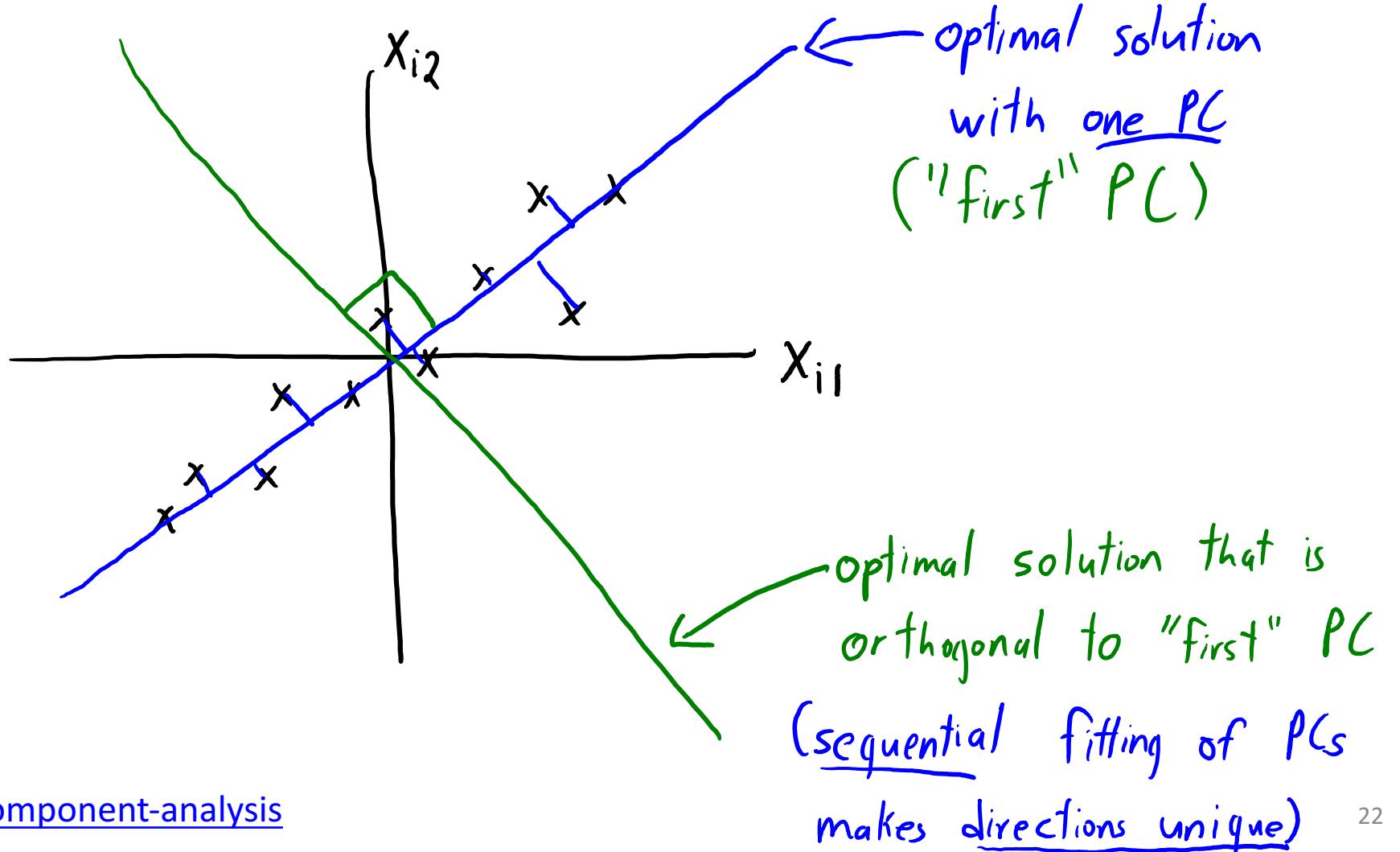


I can get 0 error
on every data point.



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

Basis, Orthogonality, Sequential Fitting



Notice that if we require $\|w_c\|=1$ then $w_c^T w_c = 1$

and if we also require orthogonality $w_c^T w_{c'} = 0$ for $c \neq c'$ then:

$$WW^T = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_k^T \end{bmatrix} \begin{bmatrix} | & | & | \\ w_1 & w_2 & \cdots & w_k \end{bmatrix} = \begin{bmatrix} w_1^T w_1 & w_1^T w_2 & \cdots & w_1^T w_k \\ w_2^T w_1 & w_2^T w_2 & \cdots & w_2^T w_k \\ \vdots & \vdots & \ddots & \vdots \\ w_k^T w_1 & w_k^T w_2 & \cdots & w_k^T w_k \end{bmatrix}$$

So finding Z simplifies: $Z = XW^T(WW^T)^{-1} = XW^T$

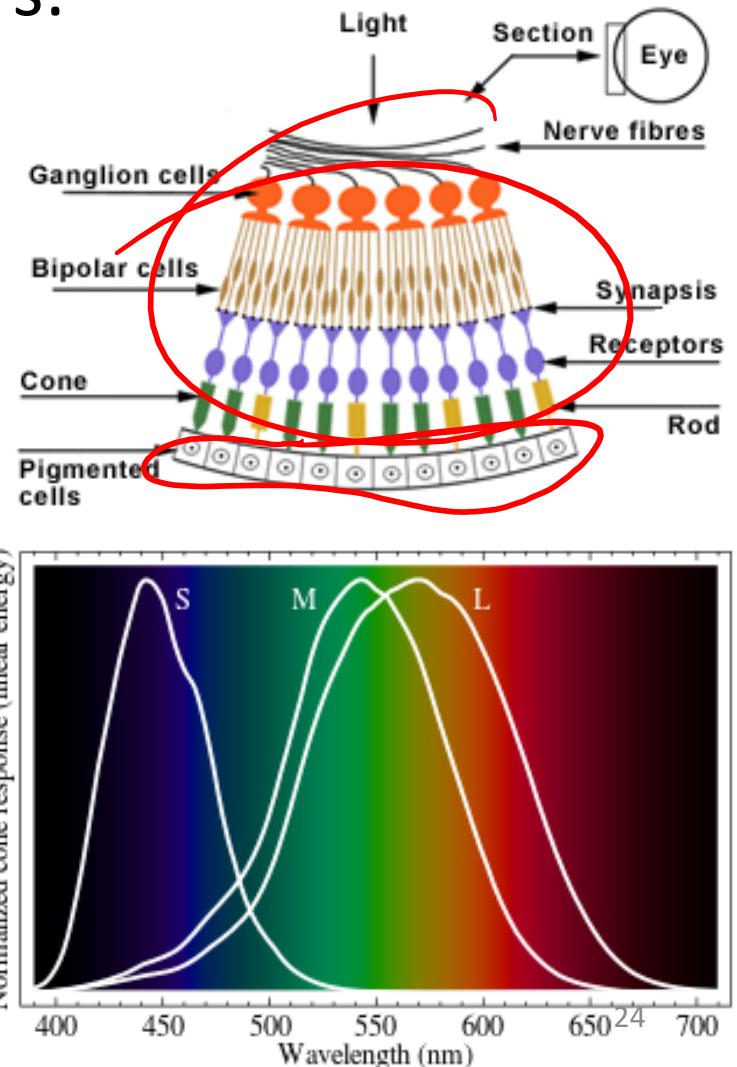
(If $k=1$ then $z_i = \frac{w_c^T x_i}{w_c^T w_c} = w_c^T x_i$)

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} = I$$

Do I need all this math?

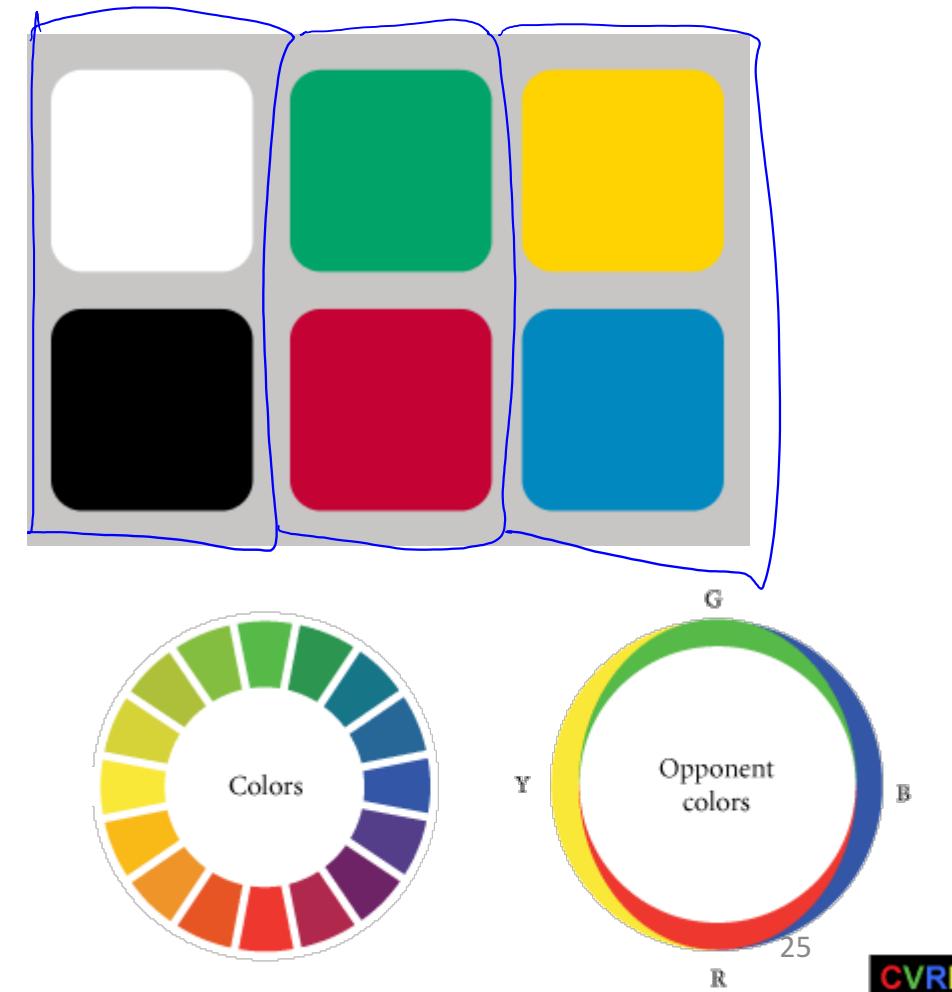
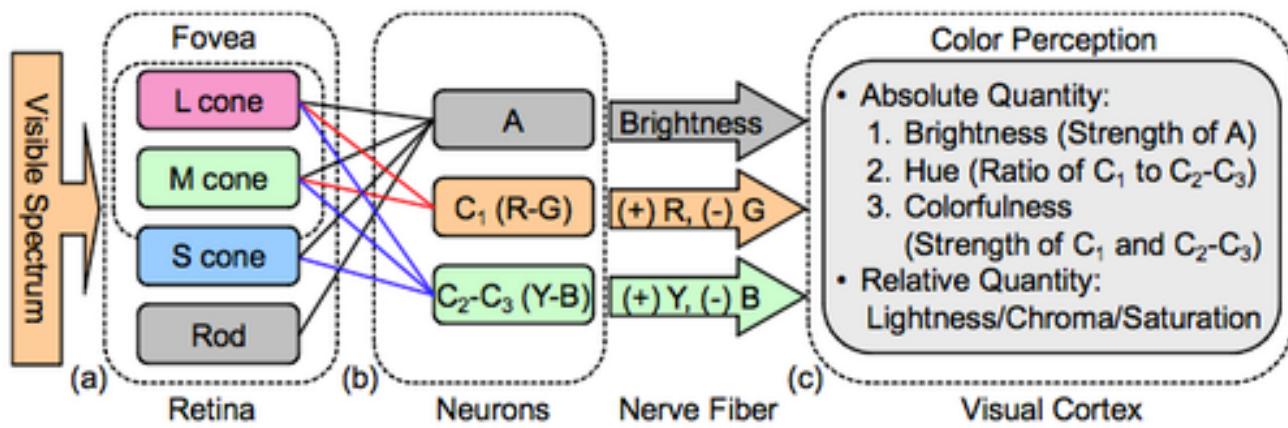
Colour Opponency in the Human Eye

- Classic model of the eye is with 4 photoreceptors:
 - Rods (more sensitive to brightness).
 - L-Cones (most sensitive to red).
 - M-Cones (most sensitive to green).
 - S-Cones (most sensitive to blue).
- Two **problems with this system**:
 - Correlation between receptors (not orthogonal).
 - Particularly between red/green.
 - We have 4 receptors for 3 colours.



Colour Opponency in the Human Eye

- Bipolar and ganglion cells seem to code using “opponent colors”:
 - 3-variable orthogonal basis:



- This is similar to PCA ($d = 4$, $k = 3$).

<http://oneminuteastronomer.com/astro-course-day-5/>

https://en.wikipedia.org/wiki/Color_vision

<http://5sensesnews.blogspot.ca/>

Colour Opponency Representation

For this pixel,
eye gets 4 signals



$$= w_1$$

First row
of W

(First PC)

↓
Analogous to means in k-means.



brightness

Can represent 4 original values with
these 3 z_i values and
matrix 'W'



red/green

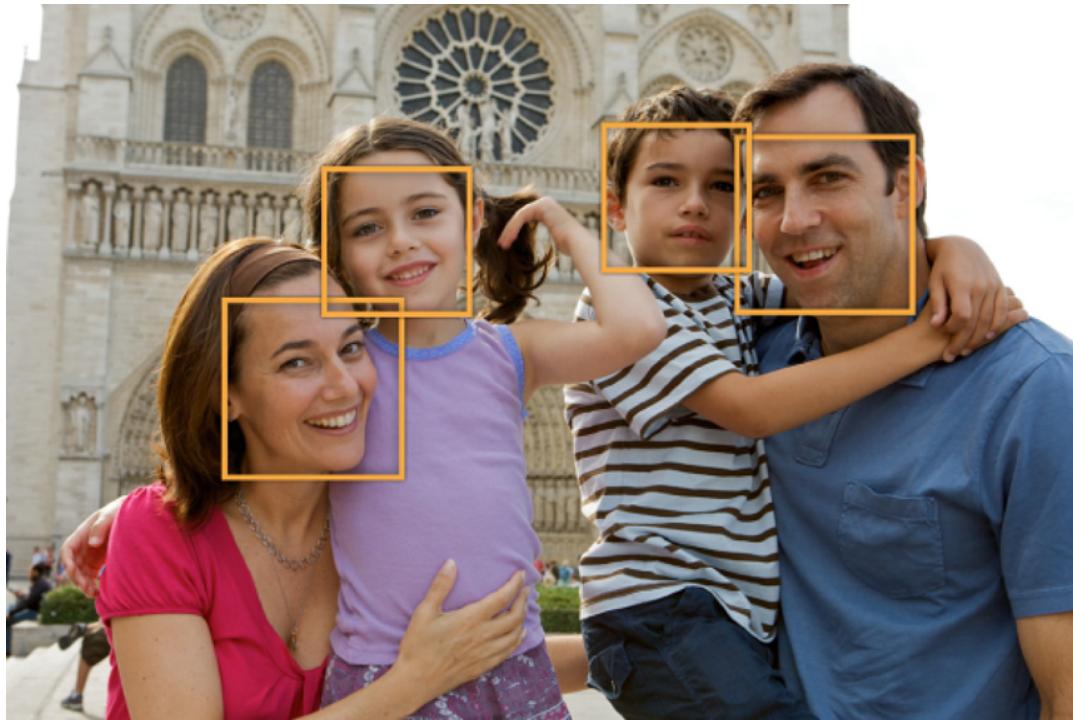
+ w_2
↓
Second
row
(4x1)



blue/yellow

Application: Face Detection

- Consider problem of face detection:



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

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 .²⁸

Eigenfaces

Compute mean μ_j of each column.



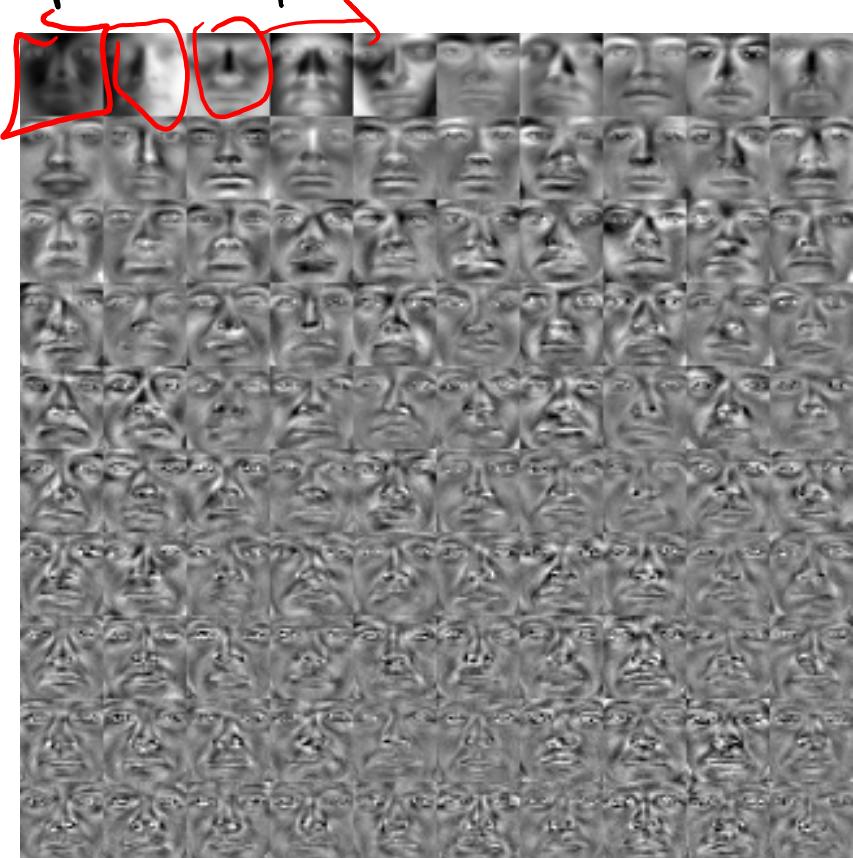
Each row of X will be pixels in one image:

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

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

Eigenfaces

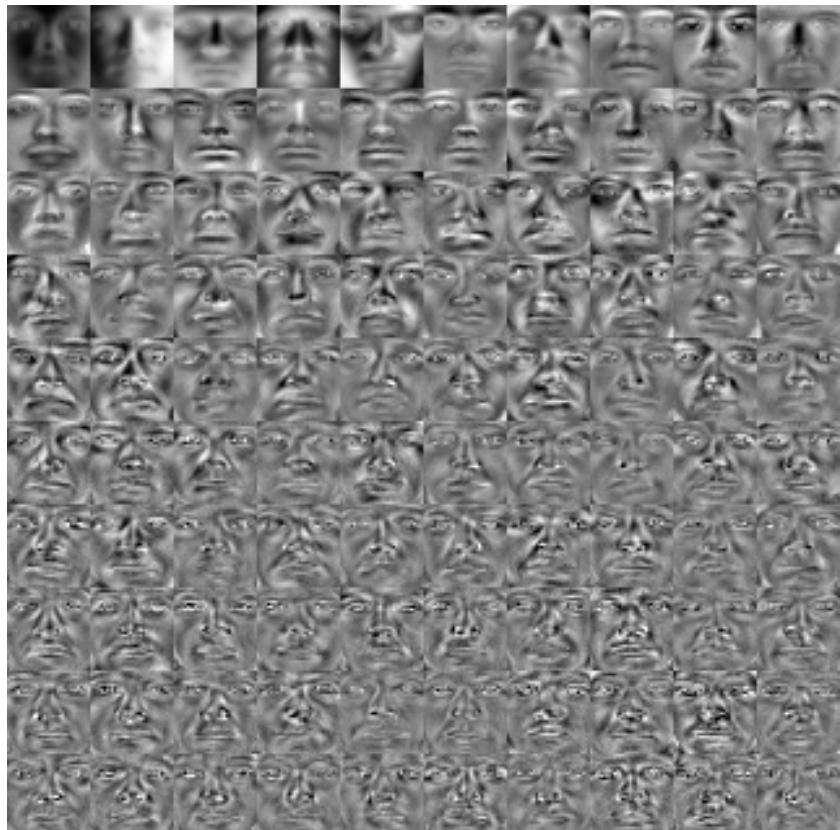
Compute top 'k' PCs on centered data: Each row of X will be pixels in one image:



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

Eigenfaces

Compute top 'k' PCs on centered data:



"Eigenface" representation:

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

x_i μ z_{i1} PC1
 $(\text{first row of } W)$

31

Eigenfaces

106 of the original faces:



"Eigenface" representation:

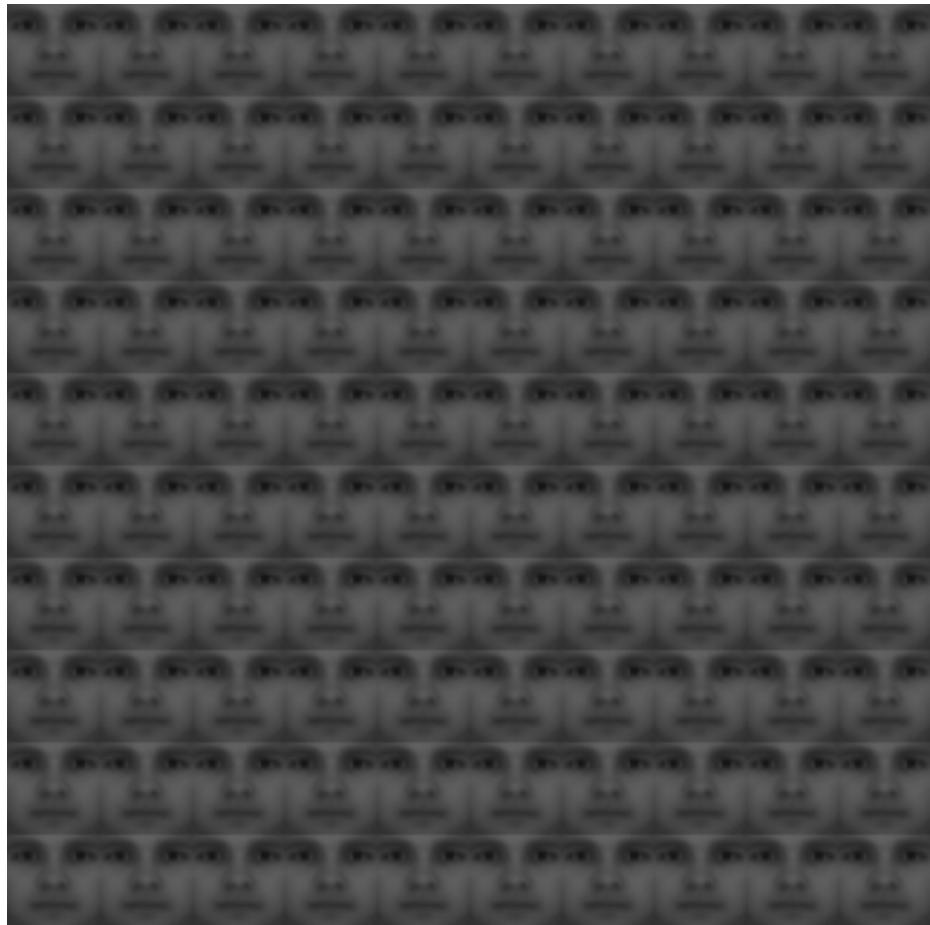
$$x_i = \mu + z_{i1} \text{PC1} + z_{i2} \text{PC2} + z_{i3} \text{PC3} + \dots$$

x_i μ z_{i1} PC1
 z_{i2} PC2
 z_{i3} PC3
 \vdots

(first row of W)

Eigenfaces

Reconstruction with $k=0$



Variance explained: 0%

"Eigenface" representation:

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

The equation illustrates the "Eigenface" representation of a face image x_i . It shows that the image x_i is equal to the mean face μ plus a linear combination of principal components (eigenfaces) PC1 , PC2 , PC3 , and so on. The coefficients z_{i1} , z_{i2} , z_{i3} , etc., represent the contribution of each eigenface to the reconstruction of the specific face x_i .

x_i μ

z_{i1} PC1
(first row of W)

z_{i2} PC2

z_{i3} PC3

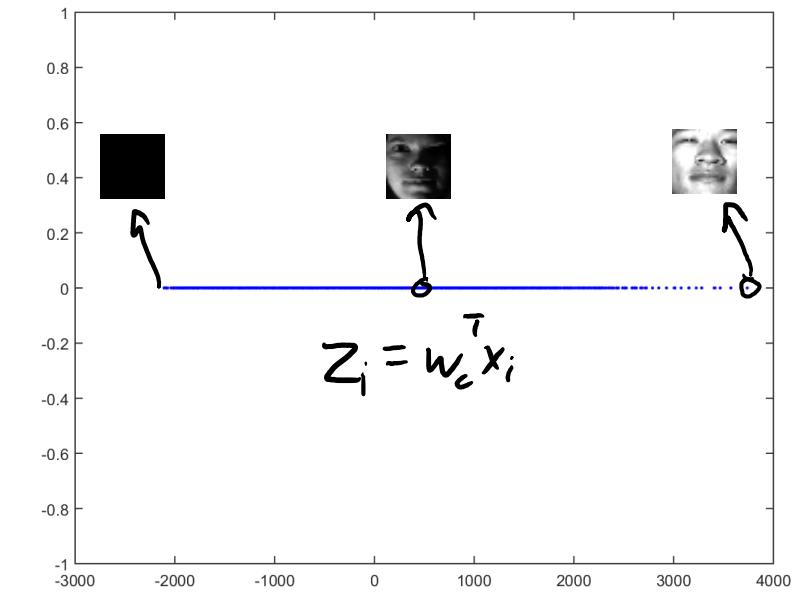
\dots

Eigenfaces

Reconstruction with $k=1$



PCA Visualization:



"Eigenface" representation:

Variance explained: 34%

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

The diagram shows the decomposition of a face image x_i into its mean μ and its projections onto the first three principal components (PC1, PC2, PC3). A red box highlights the term x_i , and another red box highlights the first row of the weight matrix w , which corresponds to the first row of the matrix P (the first row of w is labeled "(first row of w)").

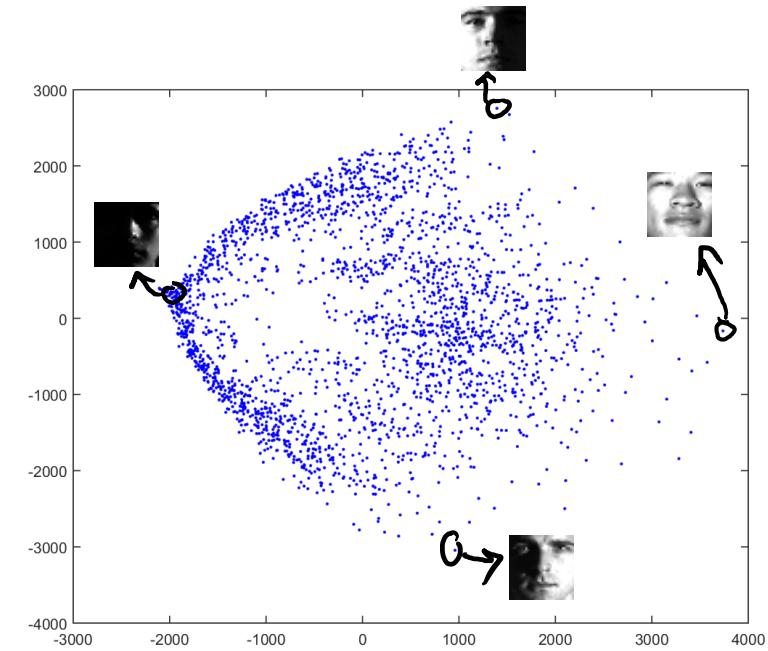
Eigenfaces

Reconstruction with $k=2$



Variance explained: 71%

PCA Visualization:



"Eigenface" representation:

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

The equation shows the decomposition of a face image x_i into its mean μ and its projections onto the first three principal component eigenvectors (Eigenfaces) PC1 , PC2 , and PC3 . The terms z_{i1} , z_{i2} , z_{i3} , and \dots represent the coefficients for each component. A red oval encloses the entire equation, highlighting the "Eigenface" representation.

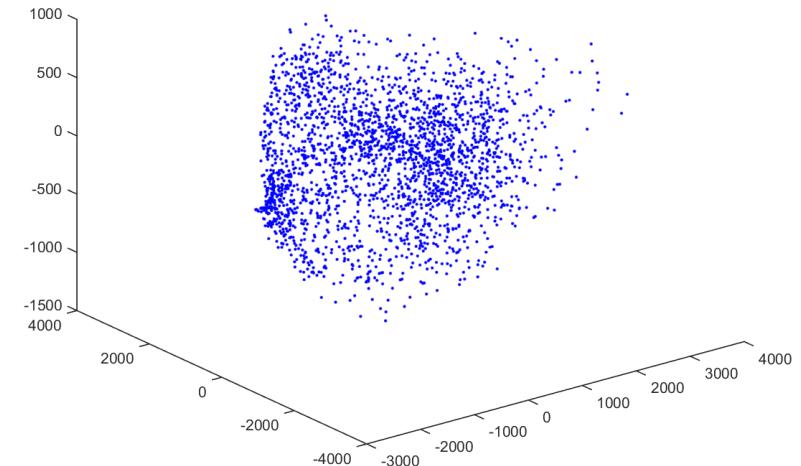
Eigenfaces

Reconstruction with $k=3$



Variance explained: 76%

PCA Visualization:



"Eigenface" representation:

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

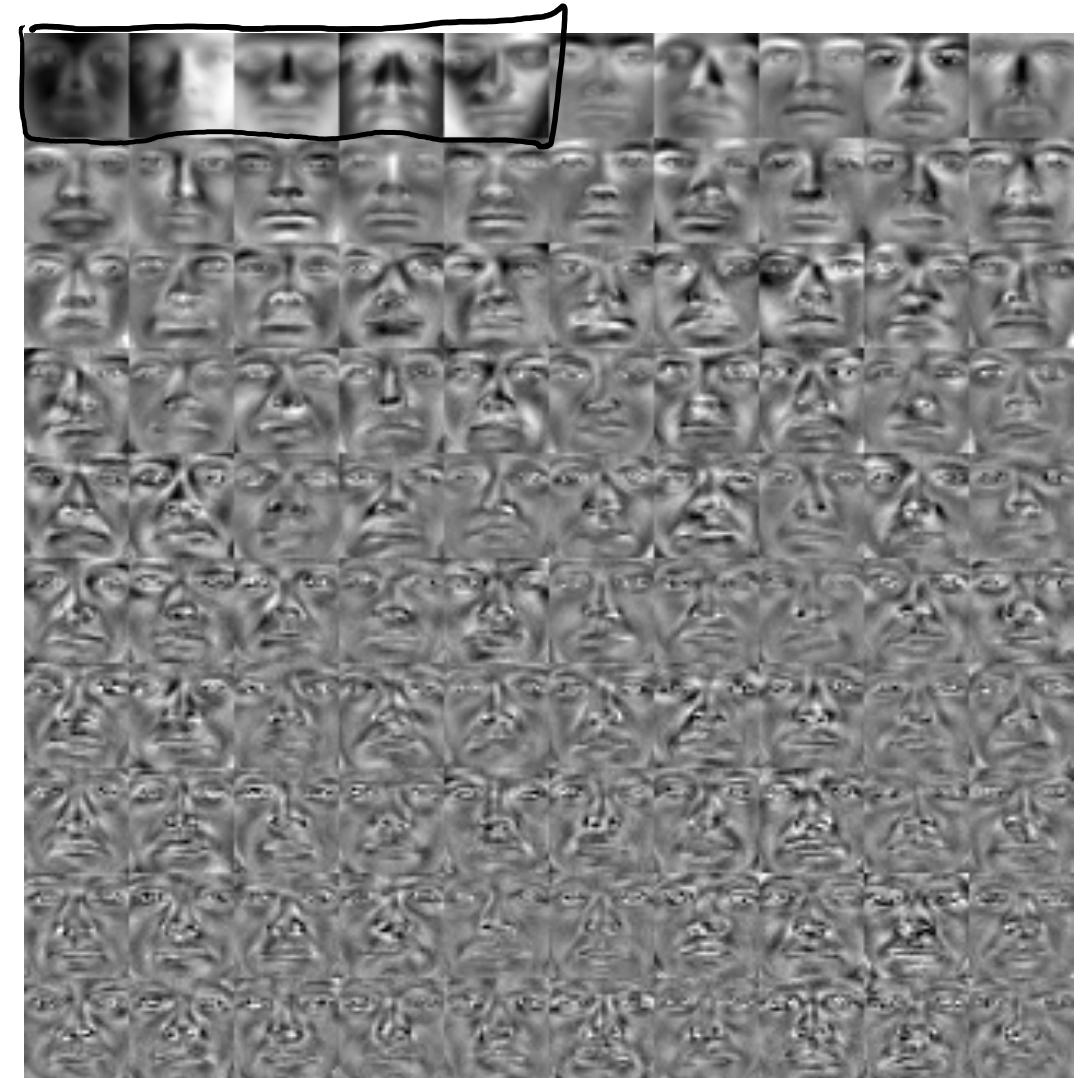
The equation shows the "Eigenface" representation of a face x_i . It consists of the mean face μ plus a linear combination of three principal components (eigenfaces) PC1 , PC2 , and PC3 , plus a residual term represented by ellipses. The first row of the weight matrix W is highlighted with a red box around the terms PC1 , PC2 , and PC3 .

Eigenfaces

Reconstruction with $k=5$



Variance explained: 80 %

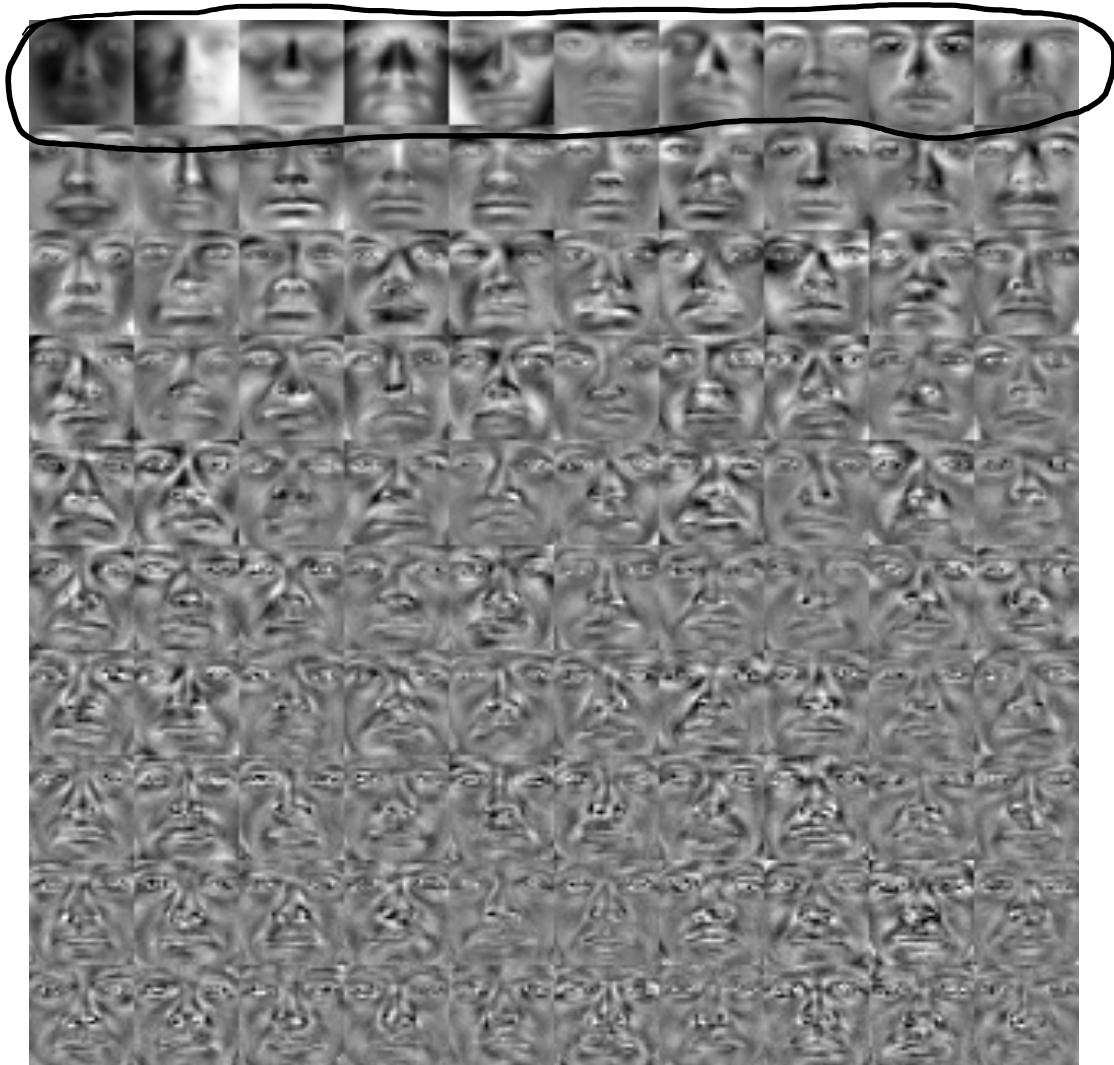


Eigenfaces

Reconstruction with $k=10$



Variance explained: 85%

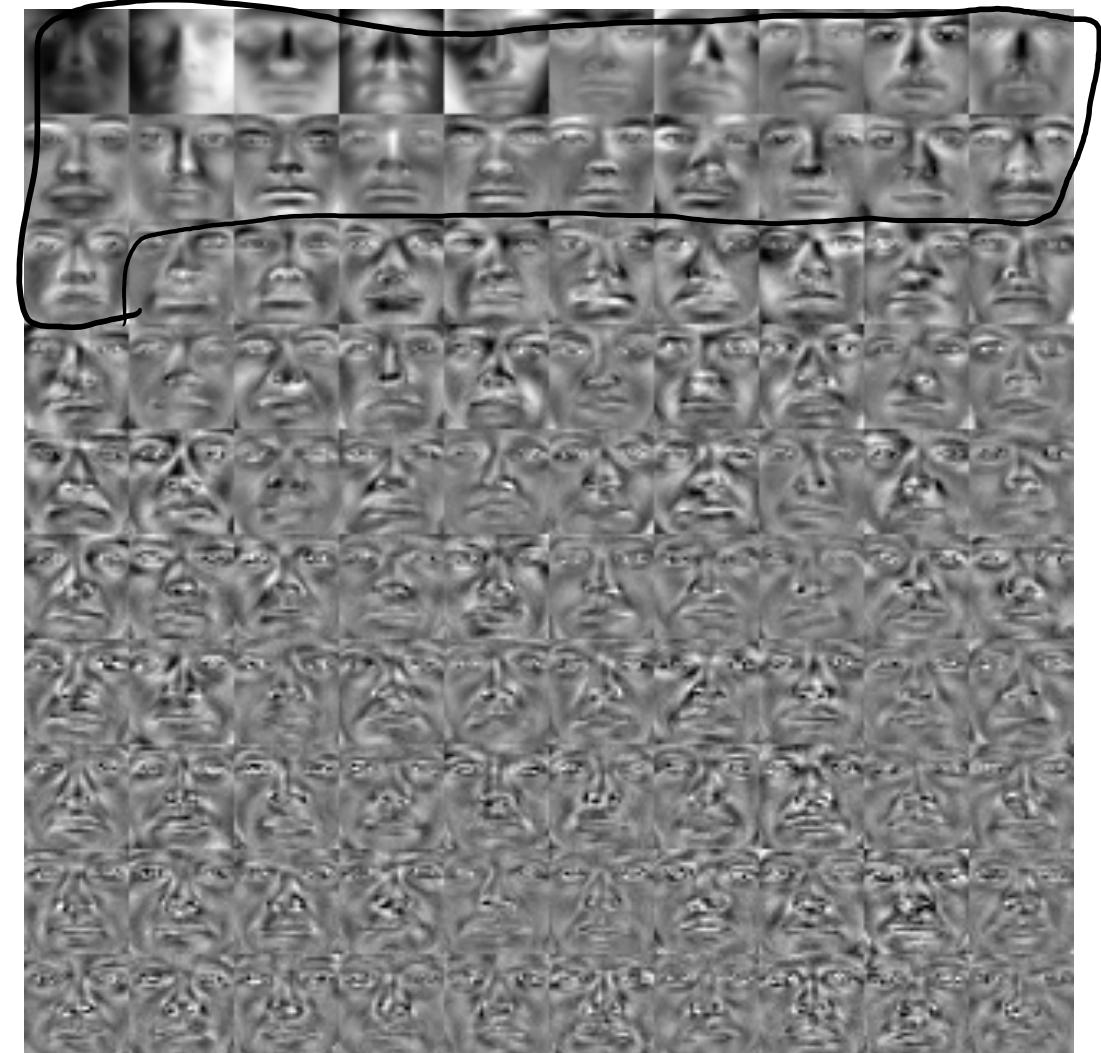


Eigenfaces

Reconstruction with $k=21$



Variance explained: 90%

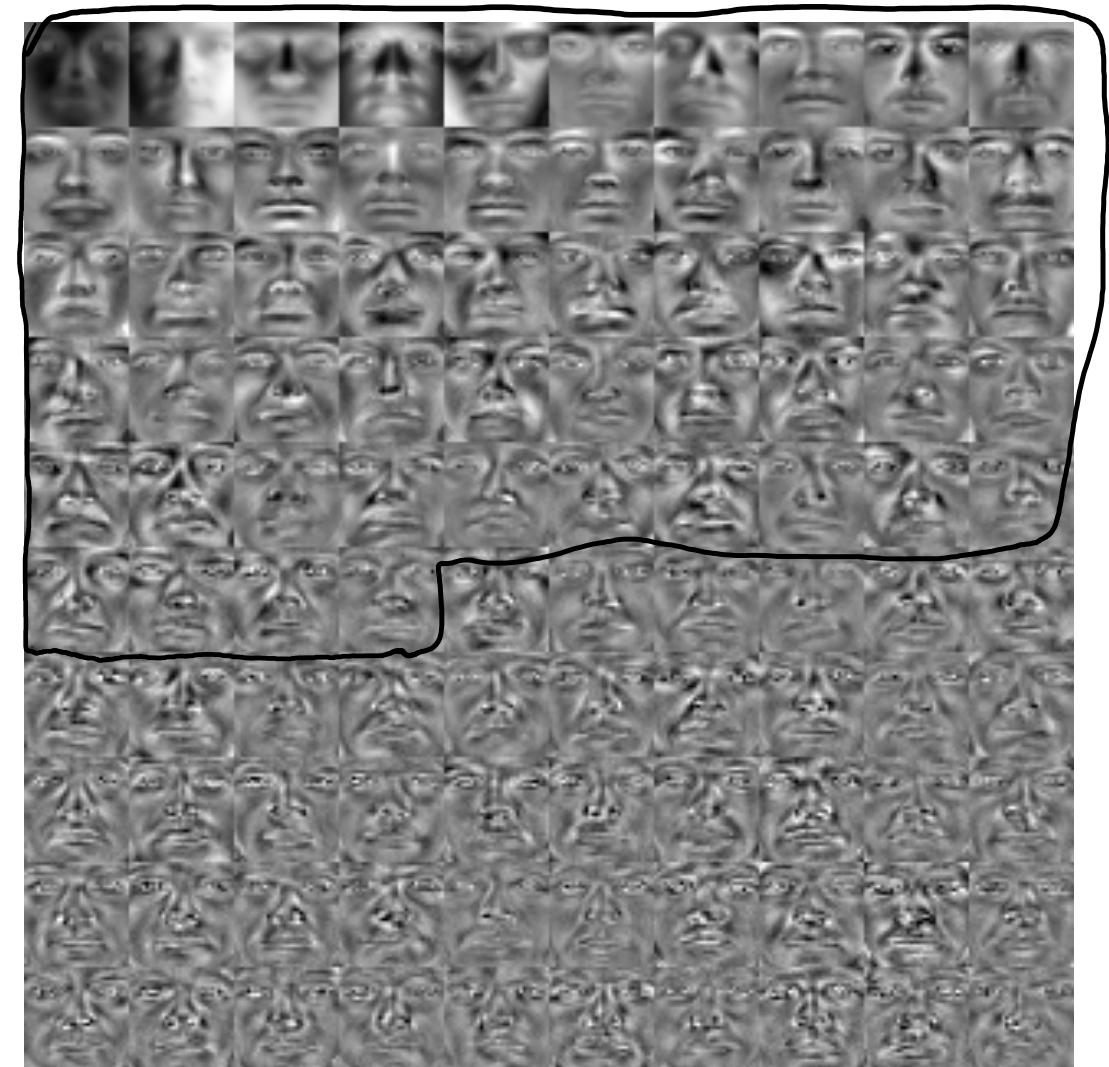


Eigenfaces

Reconstruction with $k=54$



Variance explained: 95%

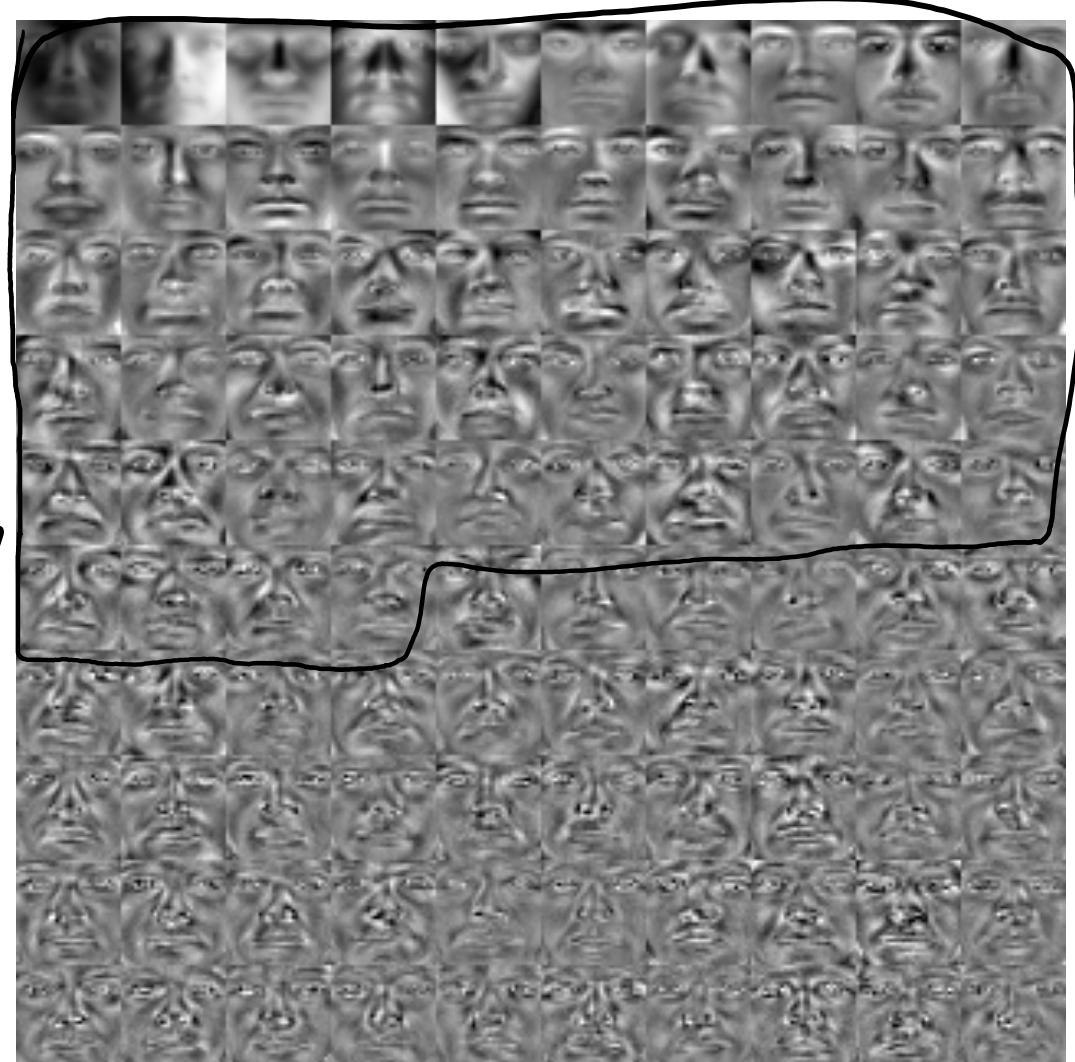


Eigenfaces

Original Images again:



Plus these
"eigenfaces"
and
the
mean,



We can replace 1024 x_i values by 54 z_i values

Summary

- **Analysis view** of PCA is that it maximizes variance.
 - We can choose ‘k’ to explain x% of the variance in the data.
- **Orthogonal basis and sequential fitting** of PCs:
 - Leads to non-redundant PCs with unique directions.
- **Biological motivation** for orthogonal and/or sparse latent factors.
- Next time: modifying PCA so it splits faces into ‘eyes’, ‘mouths’, etc.

Bonus Slide: PCA with Singular Value Decomposition

- Under constraints that $w_c^T w_c = 1$ and $w_c^T w_{c'} = 0$, use:

$$V \Sigma V^T = SVD(X)$$
$$W = V(:, 1:k)^T \quad Z = X W^T$$

- You can also quickly get compressed version of new data:

$$\hat{Z} = \hat{X} W^T$$

- If W was not orthogonal, could get Z by least squares.
- In python, `numpy.linalg.svd`