Fall 2020: Zoom@533 999 8759, pwd: mcc2020, 5:30pm-8:15pm

ECE/CS 5582 Computer Vision

Lec 14: Eigenface and Fisherface

Zhu Li

Dept of CSEE, UMKC

Office: FH560E, Email: lizhu@umkc.edu, Ph: x 2346.

http://l.web.umkc.edu/lizhu

slides created with WPS Office Linux and EqualX LaTex equation editor

Outline

- Recap:
  - SVD
  - PCA
- Face Recognition – direct learning from pixels
  - Eigenface
  - Fisherface
- Summary

SVD – projection decomposition

❑ for non square matrix: A_{mxn}:

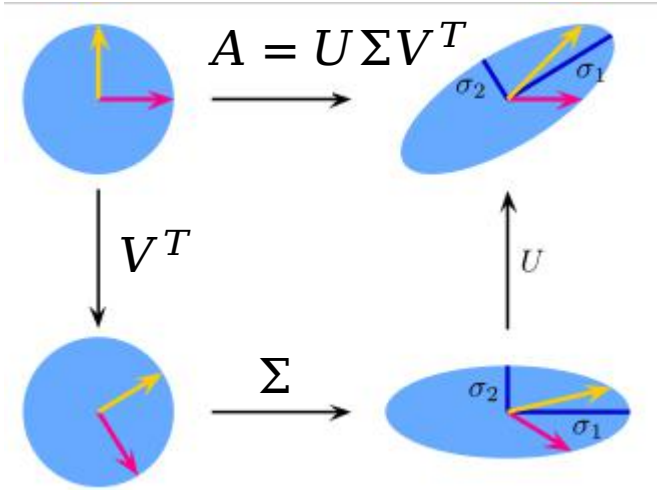
- Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$. Then a $\lambda \geq 0$ is called a *singular value* of $\mathbf{A}$, if there exist $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{v} \in \mathbb{R}^n$ such that

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{u} \quad \text{and} \quad \mathbf{A}^\mathsf{T}\mathbf{u} = \lambda \mathbf{v}$$

- We can decompose *any* matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ as

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\mathsf{T},$$

where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthonormal and Σ is a diagonal matrix of the singular values.

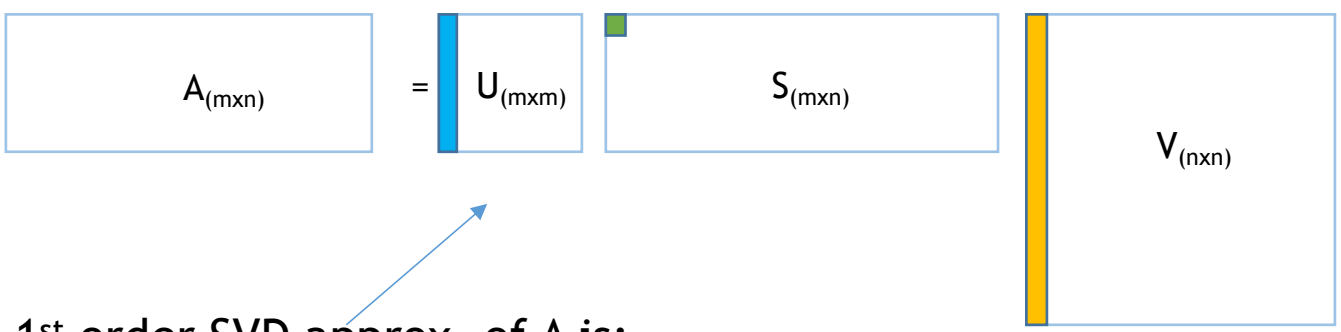


SVD as Signal Decomposition

- The Singular Value Decomposition (SVD) of an nxm matrix A, is,

$$A = USV^T = \sum_i \sigma_i u_i v_i^t$$

  - Where the diagonal of S are the eigen values of AA^T, $[\sigma_1, \sigma_2, \ldots, \sigma_n]$, called "singular values"
  - U are eigenvectors of AA^T, and V are eigen vectors of A^TA, the outer product of $u_i v_i^T$, are basis of A in reconstruction:



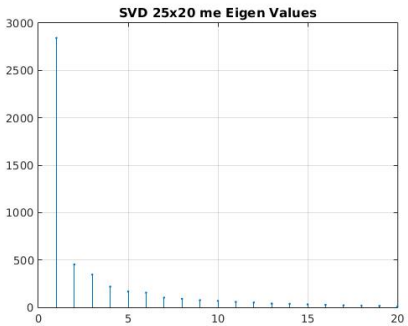
The 1st order SVD approx. of A is:

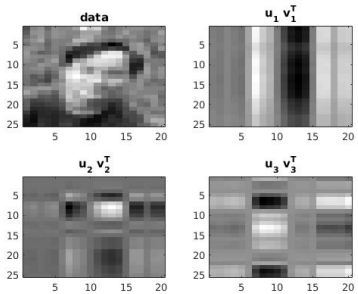
$$\sigma_1 * U(:,1) * V(:,1)^T$$

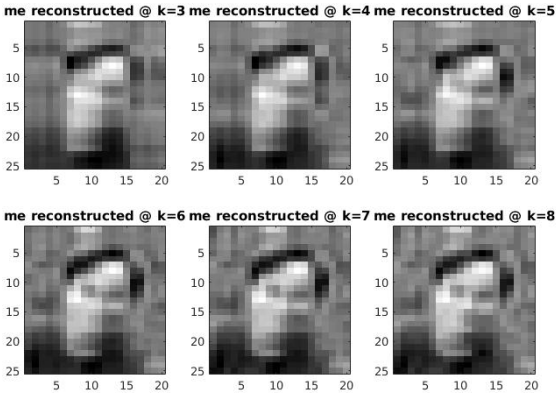
SVD approximation of an image

- Very easy…

```
function [x]=svd_approx(x0, k)
dbg=0;
if dbg
    x0= fix(100*randn(4,6));
    k=2;
end

[u, s, v]=svd(x0);
[m, n]=size(s);
x = zeros(m, n);
sgm = diag(s);

for j=1:k
    x = x + sgm(j)*u(:,j)*v(:,j)';
end
```



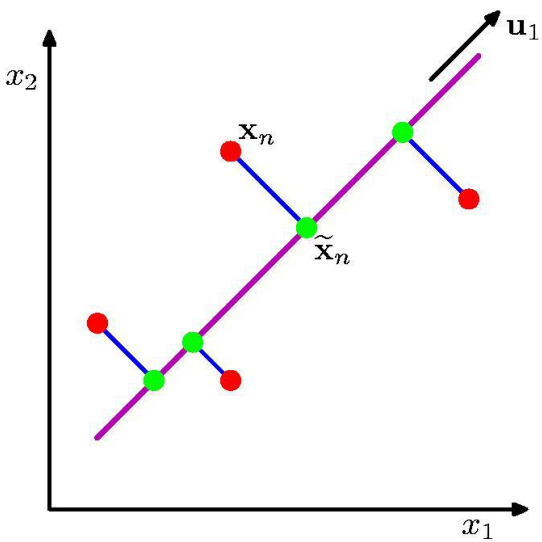




PCA- Principal Component Analysis

- Formulation:
  - Find projections, that the information/energy of the data are maximally preserved

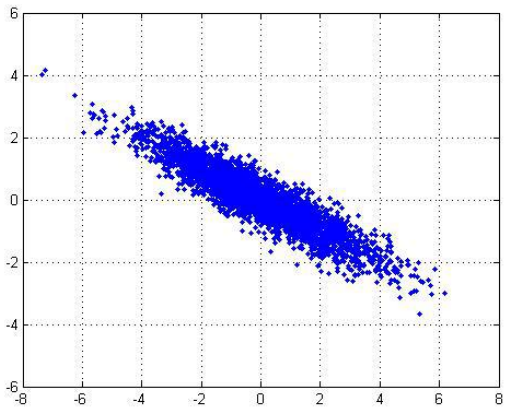
$$\max_{W} E\{x^T W^T W x\},\ s.t.,\ W^T W = I$$

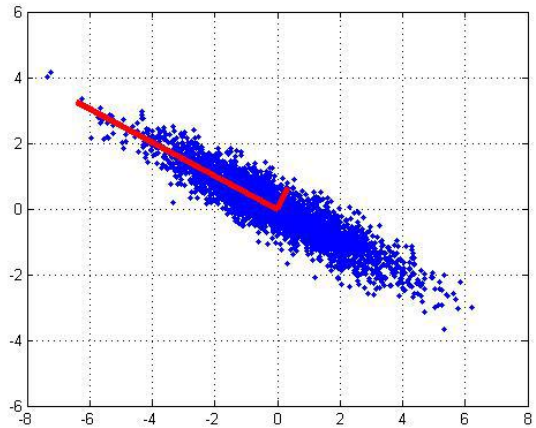


  - Matlab: [A, s, eigv]=princomp(X);

PCA Algorithm

- Center the data:
  - X = X – repmat(mean(x), [n, 1]);
- Principal component #1 points in the direction of the largest variance
- Each subsequent principal component…
  - is *orthogonal* to the previous ones, and
  - points in the directions of the largest variance of the residual subspace
- Solved by finding Eigen Vectors of the Scatter/Covarinace matrix of data:
  - S = cov(X); [A, eigv]=Eig(S)





Min Error Reconstruction Derivation of PCA Algorithm

Let $\mathbf{x} \in \mathbb{R}^N$

Let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m}$,
m: number of instances, N: dimension

Let $\mathbf{U} = \begin{pmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_N^T \end{pmatrix} \in \mathbb{R}^{N \times N}$ orthogonal matrix, $\mathbf{U}\mathbf{U}^T = \mathbf{I}_N$

$$\mathbf{y} \doteq \mathbf{U}\mathbf{x}, \ \mathbf{x} = \mathbf{U}^T\mathbf{y} = \sum_{i=1}^{N} \mathbf{u}_i y_i$$

$\hat{\mathbf{x}} \doteq \sum_{i=1}^{M} \mathbf{u}_i y_i, \ (M \leq N)$ approximation of $\mathbf{x}$ using M basis vectors only.

$\varepsilon^2 \doteq \mathbb{E}\{\|\mathbf{x} - \hat{\mathbf{x}}\|^2\} = \frac{1}{m} \sum_{j=1}^{m} ||\mathbf{x}_j - \hat{\mathbf{x}}_j||$, average error

GOAL: $\arg\min_{\mathbf{U}} \varepsilon^2$, s.t $\mathbf{U}^T\mathbf{U} = \mathbf{I}_N$

Justification of PCA Algorithm

$$\varepsilon^2 = \mathbb{E}\{\|\mathbf{x} - \hat{\mathbf{x}}\|^2\} = \mathbb{E}\{\| \sum_{i=1}^{N} \mathbf{u}_i y_i - \sum_{i=1}^{M} \mathbf{u}_i y_i \|^2\}$$

$$= \mathbb{E}\{ \sum_{i=M+1}^{N} y_i \mathbf{u}_i^T \mathbf{u}_i y_i\} = \sum_{i=M+1}^{N} \mathbb{E}\{y_i^2 \}$$

$$= \sum_{i=M+1}^{N} \mathbb{E}\{(\mathbf{u}_i^T \mathbf{x})(\mathbf{x}^T \mathbf{u}_i)\}$$

Remaining dimension

$$= \sum_{i=M+1}^{N} \mathbf{u}_i^T \mathbb{E}\{\mathbf{x}\mathbf{x}^T\} \mathbf{u}_i$$

x is centered!

$$= \sum_{i=M+1}^{N} \mathbf{u}_i^T \boldsymbol{\Sigma} \mathbf{u}_i$$

PCA – reconstruction error minimization

GOAL: $\arg \min_{\mathbf{u}_{M+1},\ldots,\mathbf{u}_N} \varepsilon^2$

Use Lagrange-multipliers for the constraints, KKT conditi

$$
\begin{aligned}
L &= \varepsilon^2 - \sum_{i=M+1}^{N} \lambda_i(\mathbf{u}_i^T\mathbf{u}_i - 1) \\
&= \sum_{i=M+1}^{N} \mathbf{u}_i^T\boldsymbol{\Sigma}\mathbf{u}_i - \sum_{i=M+1}^{N} \lambda_i(\mathbf{u}_i^T\mathbf{u}_i - 1)
\end{aligned}
$$

$$
\frac{\partial L}{\partial \mathbf{u}_i} = [2\boldsymbol{\Sigma}\mathbf{u}_i - 2\lambda_i\mathbf{u}_i] = 0
$$

Justification of PCA - Geometry Aspects

$$\frac{\partial L}{\partial \mathbf{u}_i} = [2\boldsymbol{\Sigma}\mathbf{u}_i - 2\lambda_i \mathbf{u}_i] = 0 \Rightarrow \boldsymbol{\Sigma}\mathbf{u}_i = \lambda_i \mathbf{u}_i$$

$$\Rightarrow [\mathbf{u}_i, \lambda_i] = \text{eigenvector/eigenvalue of } \boldsymbol{\Sigma}.$$

$$\boxed{\varepsilon^2 = \sum_{i=M+1}^{N} \mathbf{u}_i^T \boldsymbol{\Sigma} \mathbf{u}_i = \sum_{i=M+1}^{N} \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \sum_{i=M+1}^{N} \lambda_i}$$

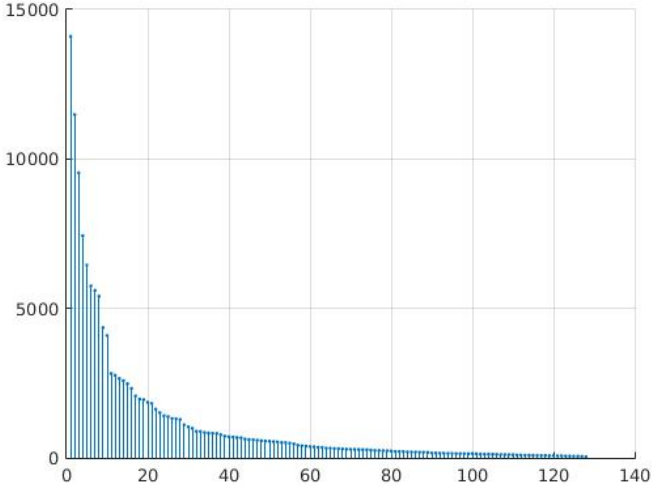
The error ε^2 is minimal

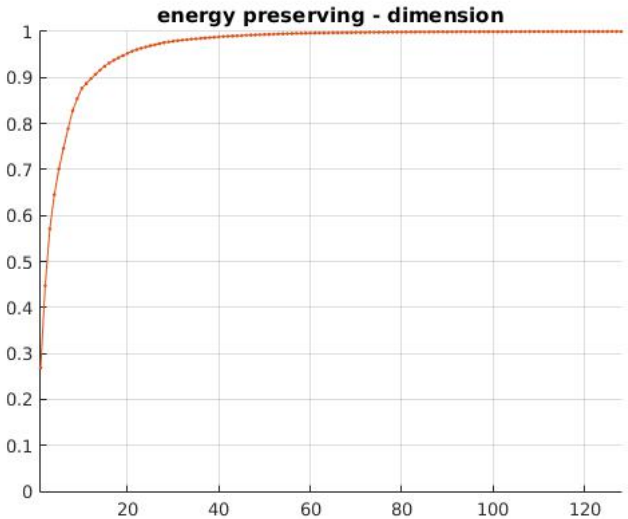
if $\lambda_{M+1}, \ldots \lambda_N$ are the smallest eigenvalues of $\boldsymbol{\Sigma}$,

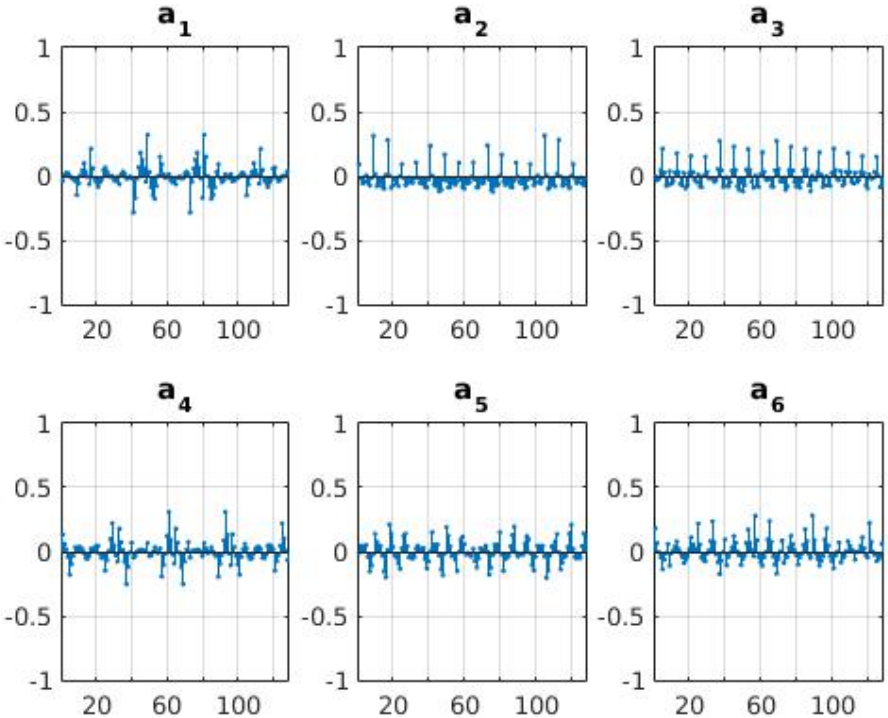
and $\mathbf{u}_{M+1}, \ldots, \mathbf{u}_N$ are the corresponding eigenvectors.

PCA for SIFT Dimension Reduction

- The PCA dimension reduction of SIFT data

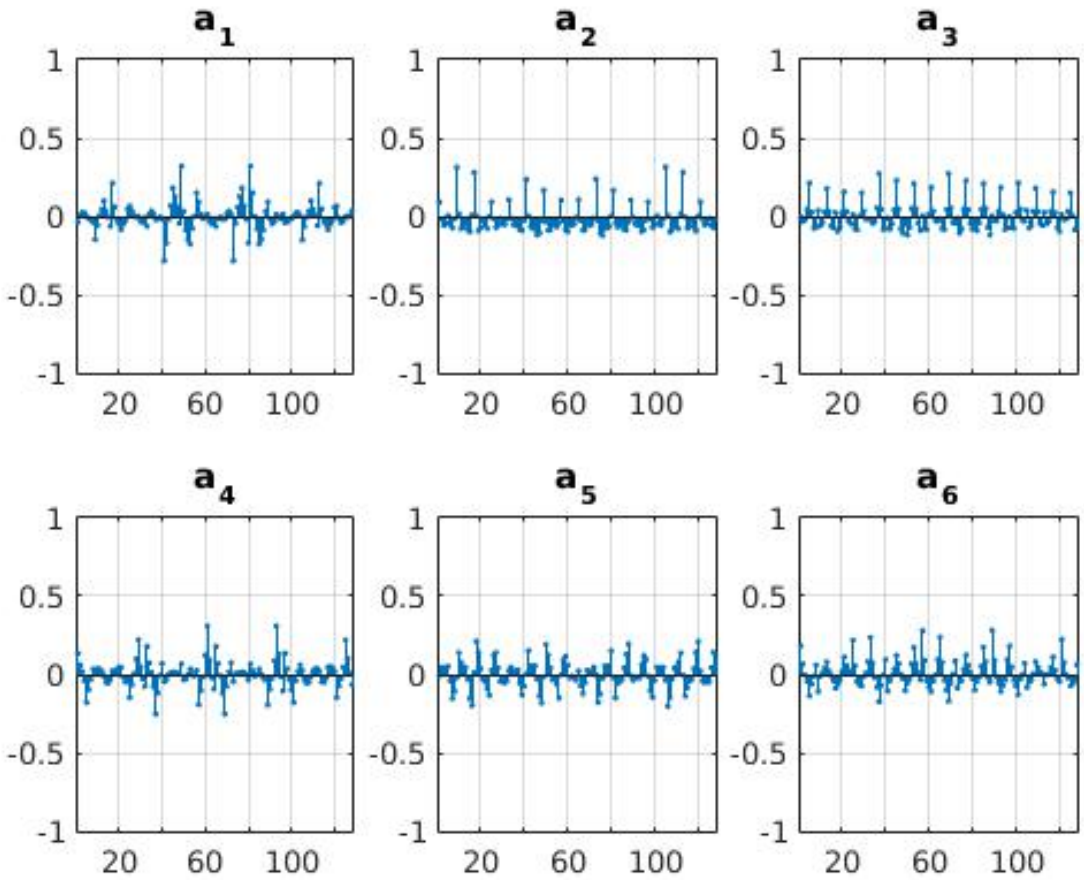






PCA SIFT Basis

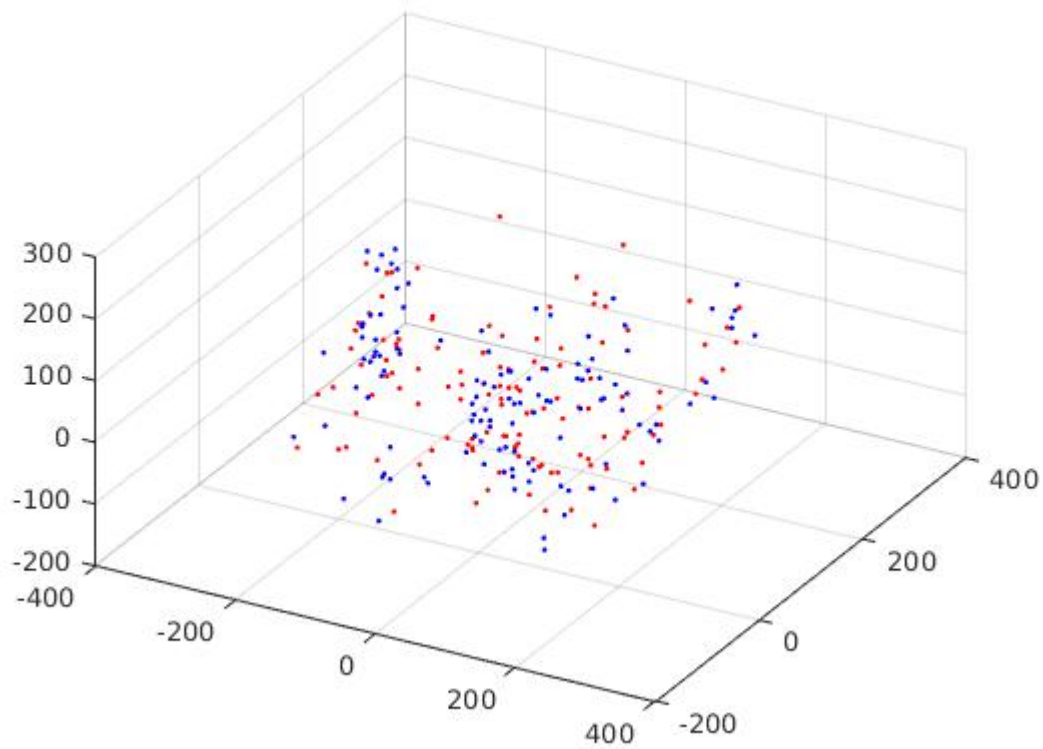
- SIFT Basis
  - A= princomp(double(sift));



SIFT Embedding via PCA

- Project SIFT to desired lower dimensional space
  - Matlab:

```
offs=gd_sift_offs(k-1)+1: gd_sift_offs(k);
sift = gd_sift_cdvs(offs,: );
x = sift*A1(:,1:kd)
% plot 3d
plot3(x(:,1), x(:,2), x(:,3), '.r');
```



Outline

- Recap:
  - SVD
  - PCA
- Face Recognition – direct learning from pixels
  - Eigenface
  - Fisherface
- Summary

The Face Recognition Problem

❑ The Verification vs Identification Problem

 =  ? Verification problem 1:1

 =  ? Identification/Recognition problem 1:N

Holistic Approach: Treat Pixel Directly as features

1. Treat pixels as a vector

h

w

X in R^{wxh}

2. Recognize face by nearest neighbor

$\{Y_1, Y_2, Y_3, \ldots, Y_n\}$ in R^{wxh}

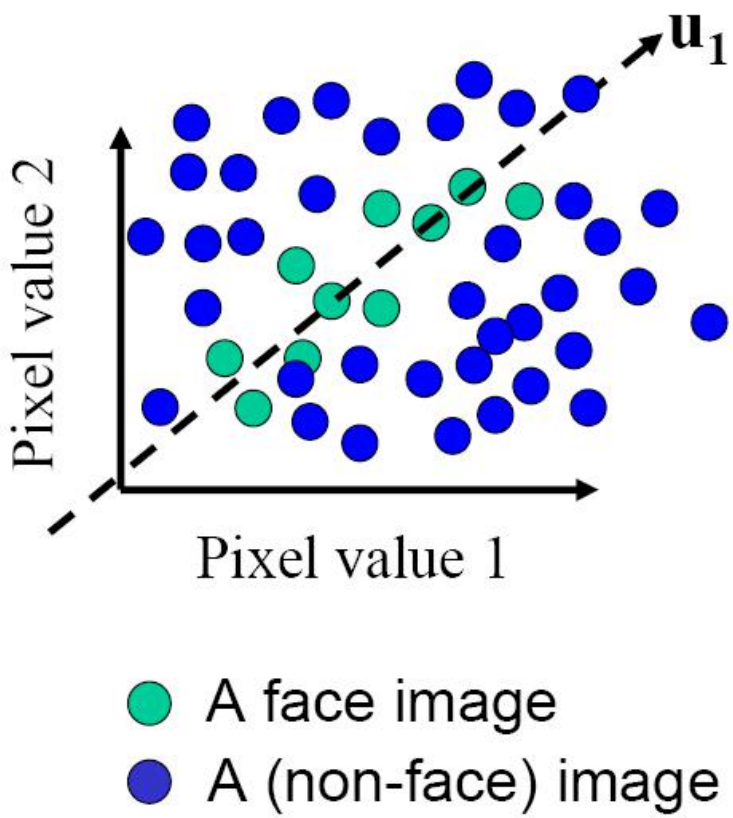
$$k = \underset{k}{\operatorname{argmin}} \left\| \mathbf{y}_k^T - \mathbf{x} \right\|$$

The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 120x120 pixel image = 14400 dimensions
  - Slow and lots of storage
- But very few 14,400-dimensional vectors are valid face images
- We want to effectively model the subspace of face images

The space of all face images

- Eigenface idea: construct a low-dimensional linear subspace that best explains the variation/scatter of the face images



Principal Component Analysis (PCA)

- Given: N data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in R^d
- We want to find a new set of features that are linear combinations of original ones:

$$u(\mathbf{x}_i) = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$

- ($\boldsymbol{\mu}$: mean of data points)
- Choose unit vector $\mathbf{u}$ in R^d that captures the most data variance

Principal Component Analysis

- Direction that maximizes the variance of the projected data:

$$\text{Maximize} \quad \frac{1}{N}\sum_{i=1}^{N} \underbrace{\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu)}_{\text{Projection of data point}}(\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu))^{\mathrm{T}}$$

subject to ||**u**||=1

$$= \mathbf{u}^{\mathrm{T}}\underbrace{\left[1/N\sum_{i=1}^{N}(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^{\mathrm{T}}\right]}_{\text{Covariance matrix of data}}\mathbf{u}$$

$$= \mathbf{u}^{\mathrm{T}}\Sigma\mathbf{u}$$

The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ

Eigenfaces (PCA on face images)

1. Compute covariance matrix of face images

2. Compute the principal components ("eigenfaces")
   - K eigenvectors with largest eigenvalues

3. Represent all face images in the dataset as linear combinations of eigenfaces
   - Perform nearest neighbor on these coefficients

M. Turk and A. Pentland,
Face Recognition using Eigenfaces, CVPR 1991

Eigenfaces example

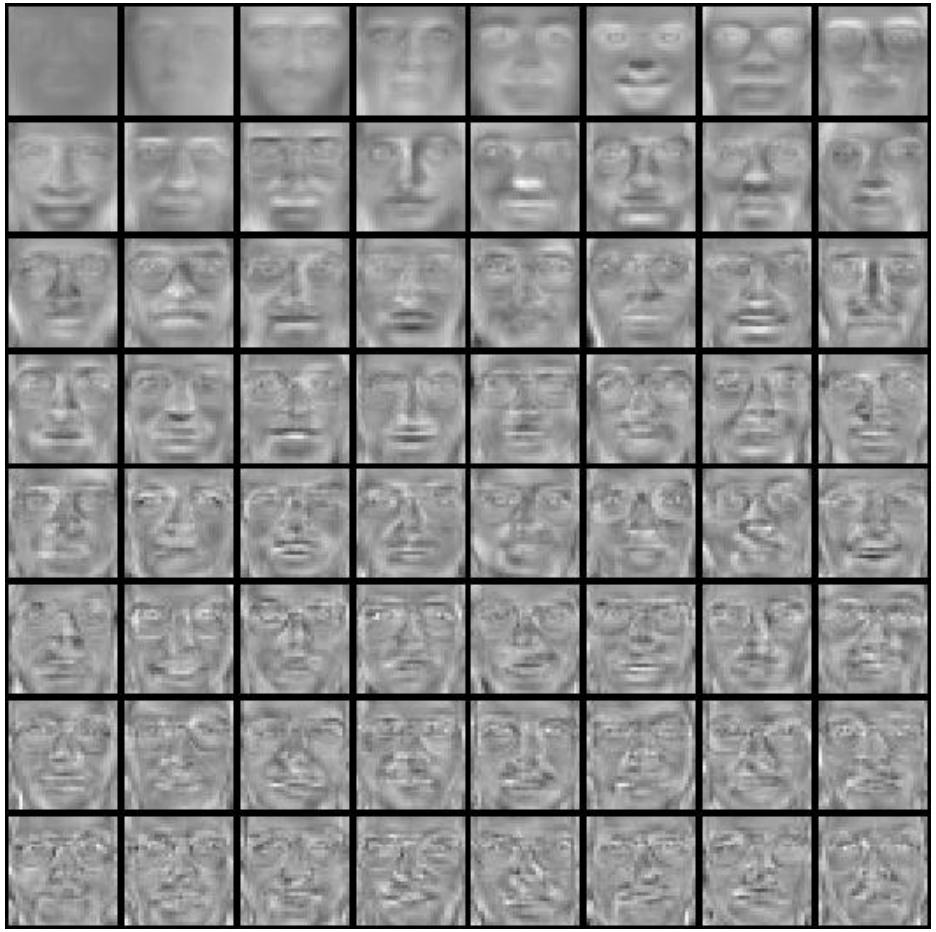
- Training images
- $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Eigenfaces example

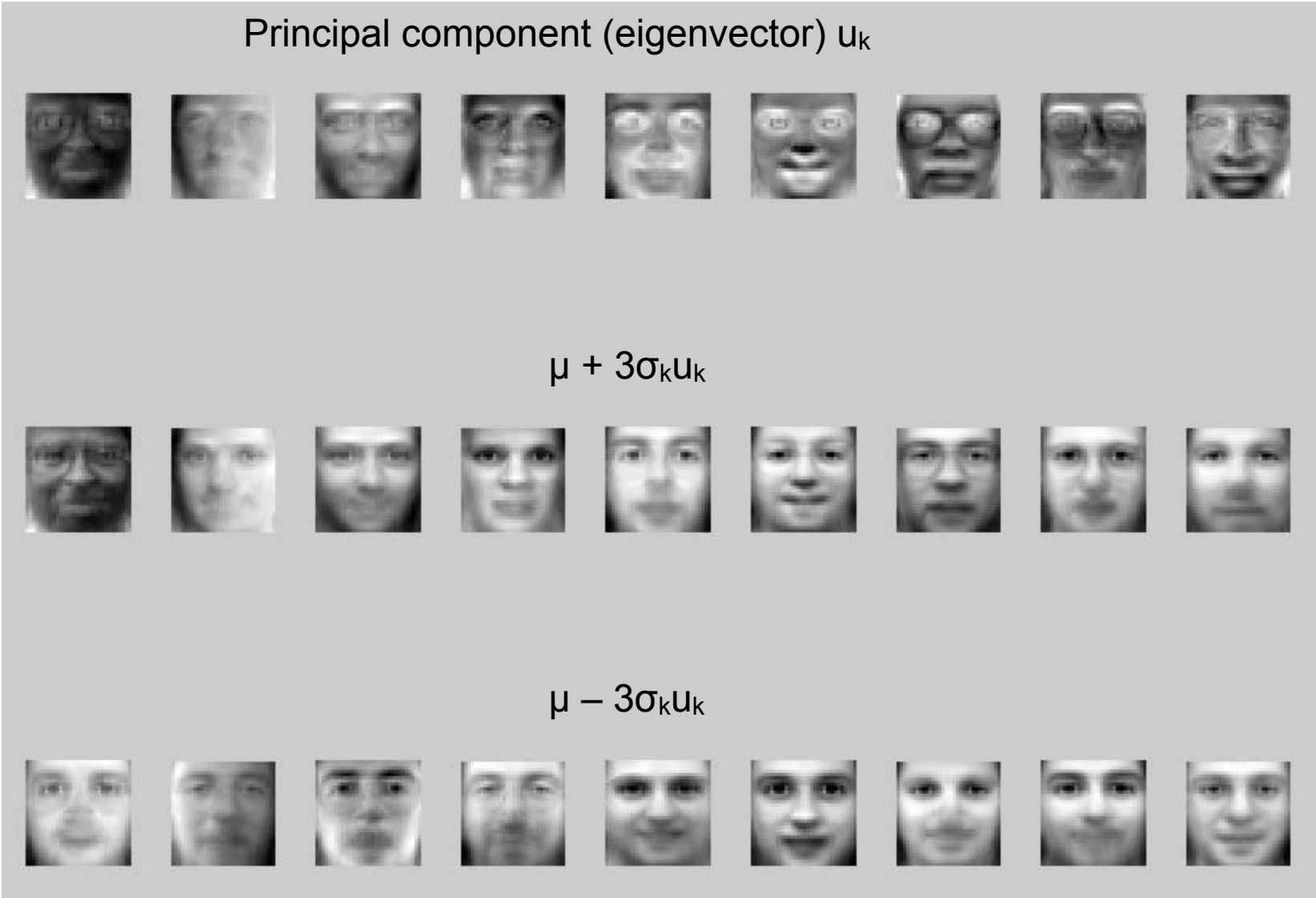
- mean face and eigen faces

Mean: μ

Top k eigenvectors: $u_1,\ldots u_k$



Visualization of eigenfaces

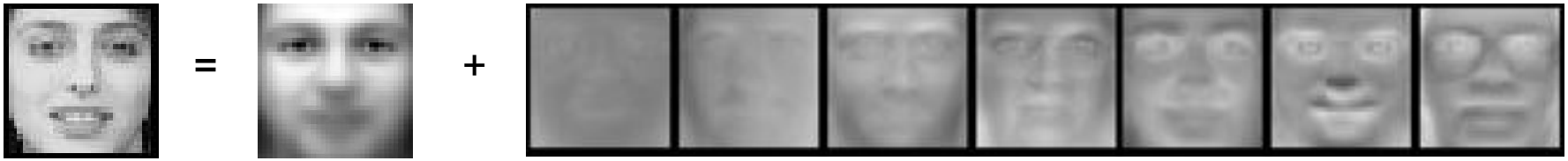


Representation and reconstruction

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$
$$= \quad w_1, \ldots, w_k$$

- Reconstruction:



$\hat{x} = \mu + w_1u_1+w_2u_2+w_3u_3+w_4u_4+ \ldots$

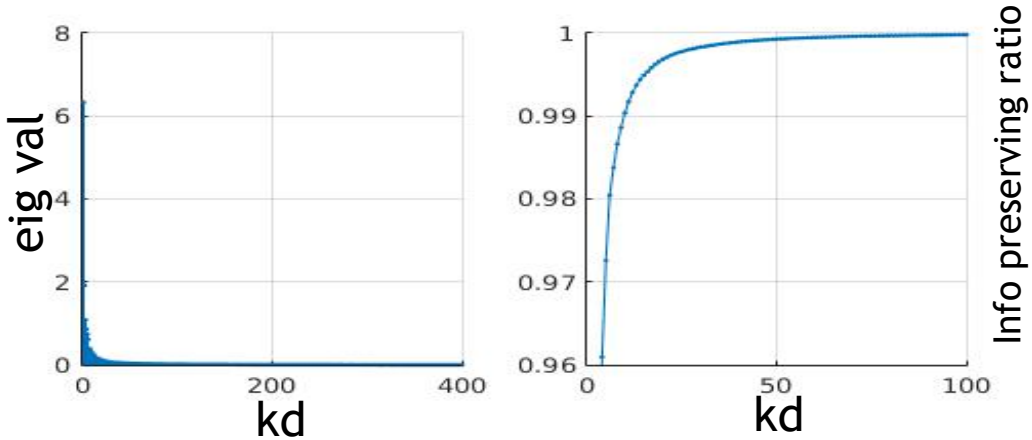
Matlab Implementation

- Scaling the face image to thumbnail size
  - Justification: as long as human eye can recognize, that contains sufficient info
  - Scale space characteristic response for face recognition
- Compute the PCA over the vectorized face data

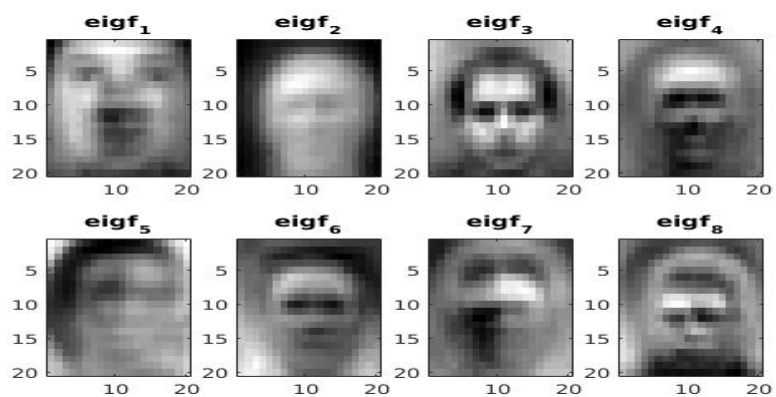
```
load faces-ids-n6680-m417-20x20.mat;
[A, s, lat]=princomp(faces);

h=20; w=20;

figure(30);
subplot(1,2,1); grid on; hold on; stem(lat, '.');
f_eng=lat.*lat;
subplot(1,2,2); grid on;
hold on; plot(cumsum(f_eng)/sum(f_eng), '.-');
```

Eigenface

- Holistic approach: treat an hxw image as a point in R^{hxw}:
  - Face data set: 20x20 face icon images, 417 subjects, 6680 face images
  - Notice that all the face images are not filling up the R^{20x20} space:

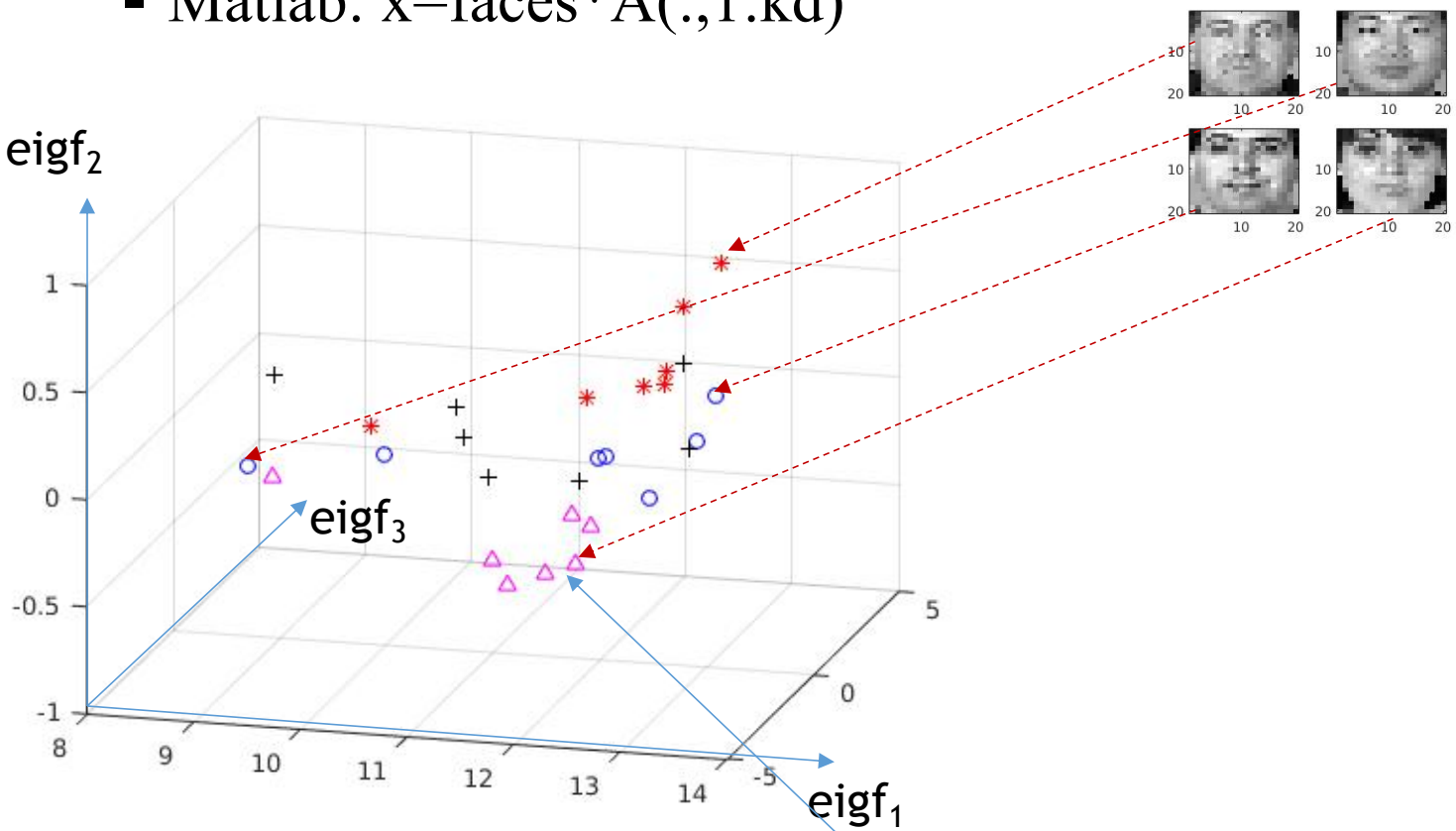


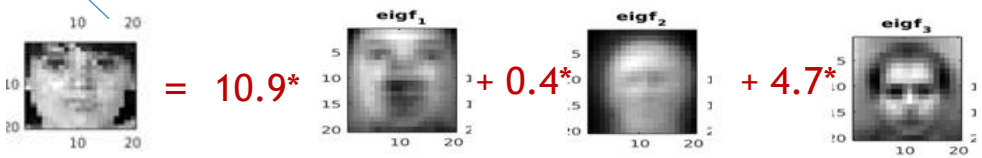
  - Eigenface basis: imagesc(reshape(A1(:,1), [h, w]));



Eigenface Projections

- Project face images to Eigenface space
  - Matlab: x=faces*A(:,1:kd)

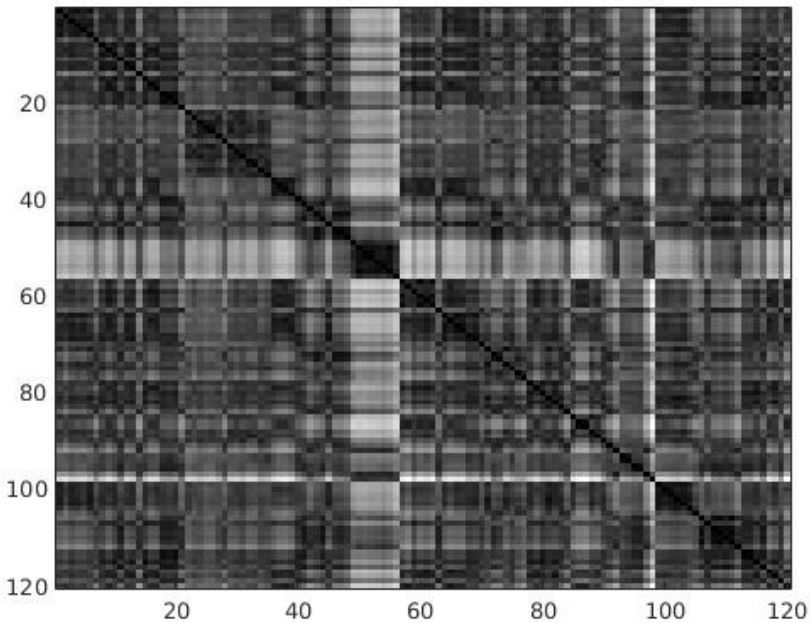


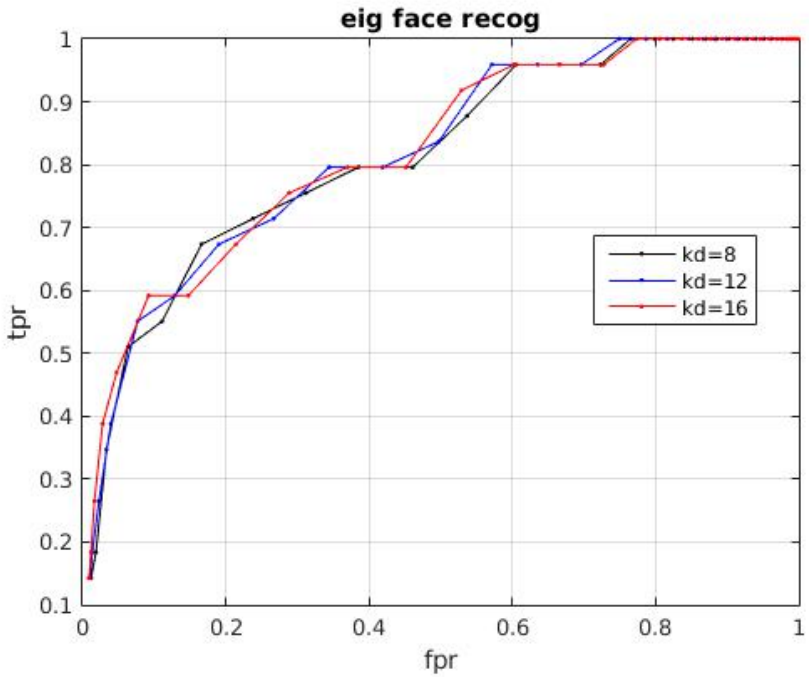


Verification Performance

- ROC of Eigen face kd=8, 12, 16

18 unique ids, 120 face images





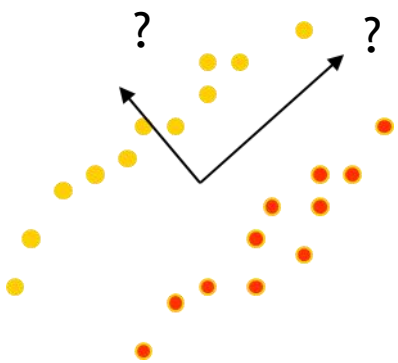
Eigenface Matlab

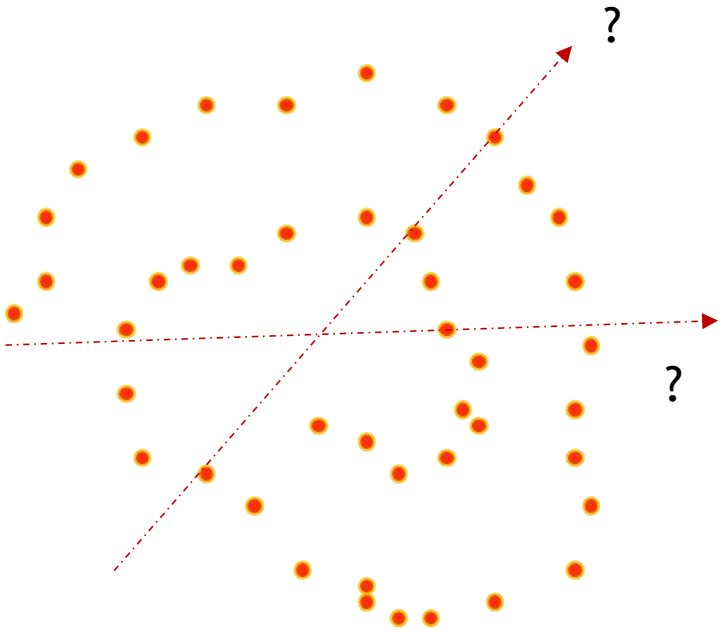
❑ Matlab code

```
% PCA:
[A, s, eigv]=princomp(faces);
kd=16; nface=120;
% eigenface projectio
x=faces*A(:, 1:kd);
f_dist = pdist2(x(1:nface,:), x(1:nface,:));
figure(32);
imagesc(f_dist); colormap('gray');

figure(33); hold  on; grid on;
d0 = f_dist(1:7,1:7); d1=f_dist(8:end, 8:end);
[tp, fp, tn, fn]= getPrecisionRecall(d0(:), d1(:), 40);
plot(fp./(tn+fp), tp./(tp+fn), '.-r', 'DisplayName', 'tpr-fpr color,
data set 1');
xlabel('fpr'); ylabel('tpr'); title('eig face recog');
legend('kd=8', 'kd=12', 'kd=16', 0);
```

Limitation of PCA

- What PCA does best ?
  - Label agnostic data compression/dimension reduction
  - Suppress noises
  - It is a linear rotation (note that $A*A^T = I$), minimum amount of distortion to the data
  - Help classification (sort of)
- What PCA can not
  - Can not fit data that is not linear (can try kernelized PCA)
  - Does not taking labelling into consideration (Fisherface !)





Outline

- Recap:
  - SVD
  - PCA
- Face Recognition – direct learning from pixels
  - Eigenface
  - Fisherface
- Summary

Why is Face Recognition Hard?

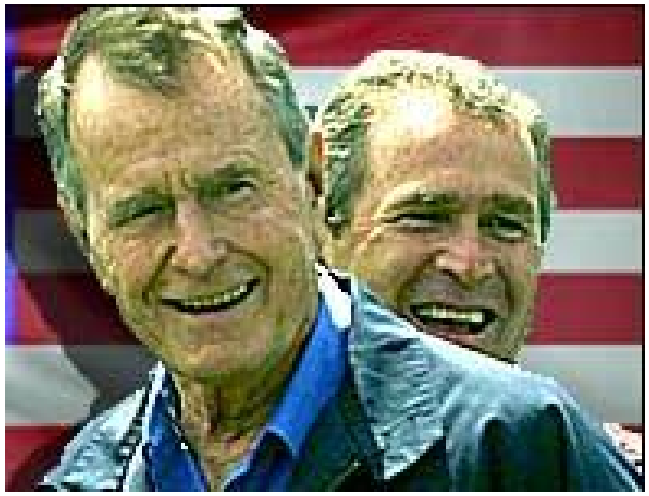
Many faces of Madonna the bad girl…

Inter-class similarities

- Different persons may have very similar appearance

www.marykateandashley.com

Twins

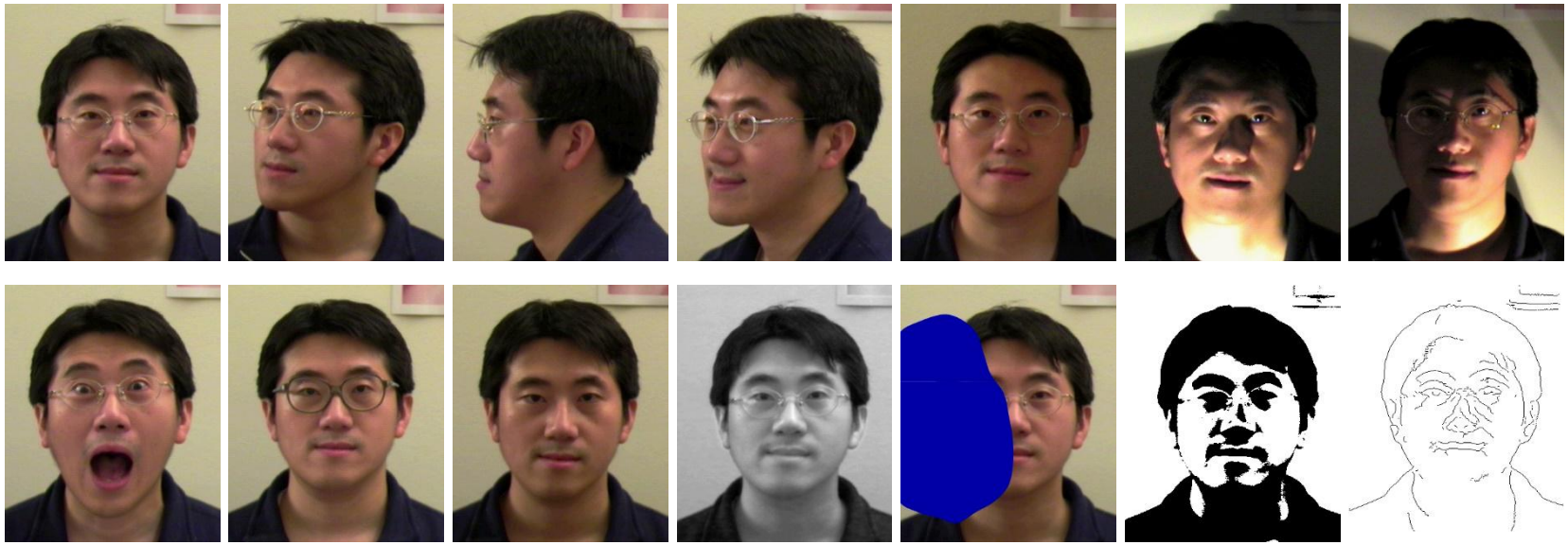


news.bbc.co.uk/hi/english/in_depth/americas/2000/us_elections

Father and son

Intra-Class Variations

- Faces with intra-subject variations in pose, illumination, expression, accessories, color, occlusions, and brightness



Face Recognition Challenges

- Identify similar faces (inter-class similarity)
- Accommodate intra-class variability due to:
  - head pose
  - illumination conditions
  - expressions
  - facial accessories
  - aging effects
  - Cartoon/sketch

Fisherface solution

- Ref:

  P. Belhumeur, J. Hespanha, D. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, PAMI, July 1997, pp. 711--720.

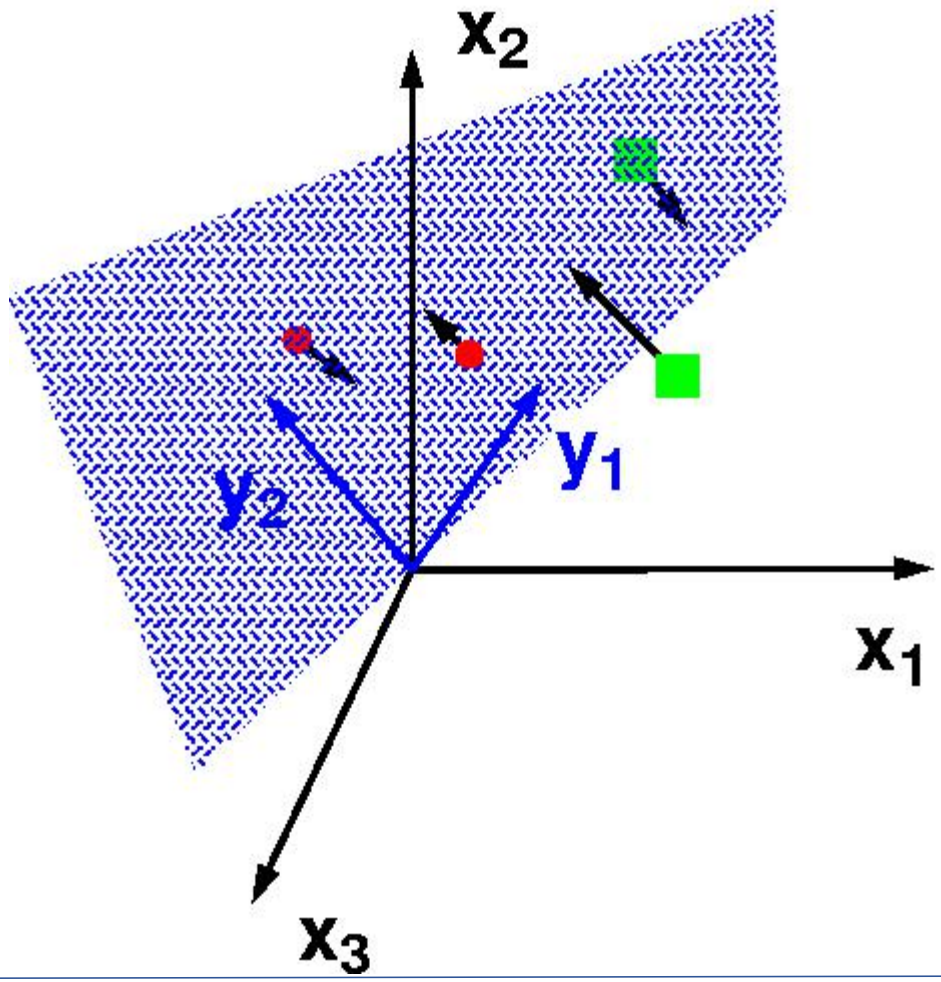
- An n-pixel image $\mathbf{x} \in \mathbf{R}^n$ can be projected to a low-dimensional feature space $\mathbf{y} \in \mathbf{R}^m$ by

$$\mathbf{y} = W\mathbf{x}$$

where W is an n by m matrix.

- Recognition is performed using nearest neighbor in $\mathbf{R}^m$.

- How do we choose a good W?



PCA & Fisher's Linear Discriminant

- Between-class scatter

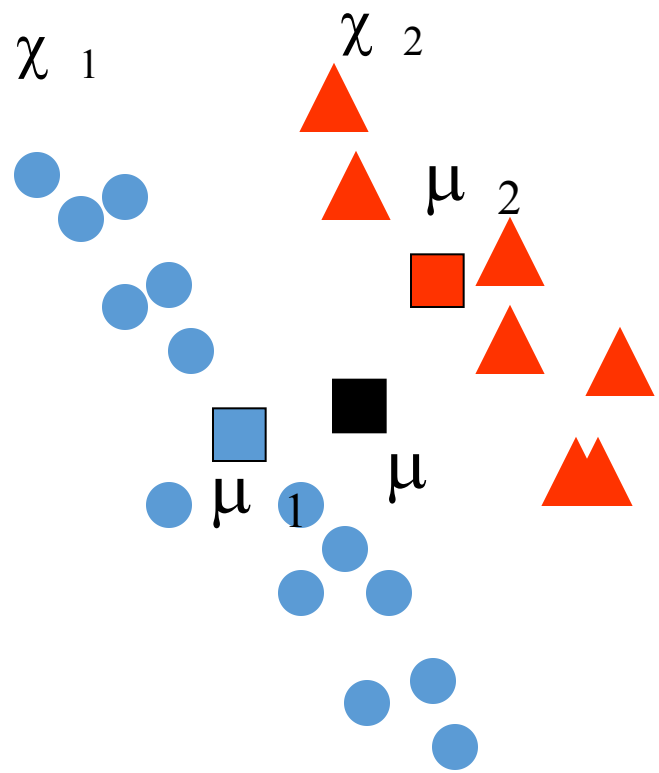
$$S_B = \sum_{i=1}^{c} |\chi_i| (\mu_i - \mu)(\mu_i - \mu)^T$$

- Within-class scatter

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu_i)(\mu_k - \mu_i)^T$$

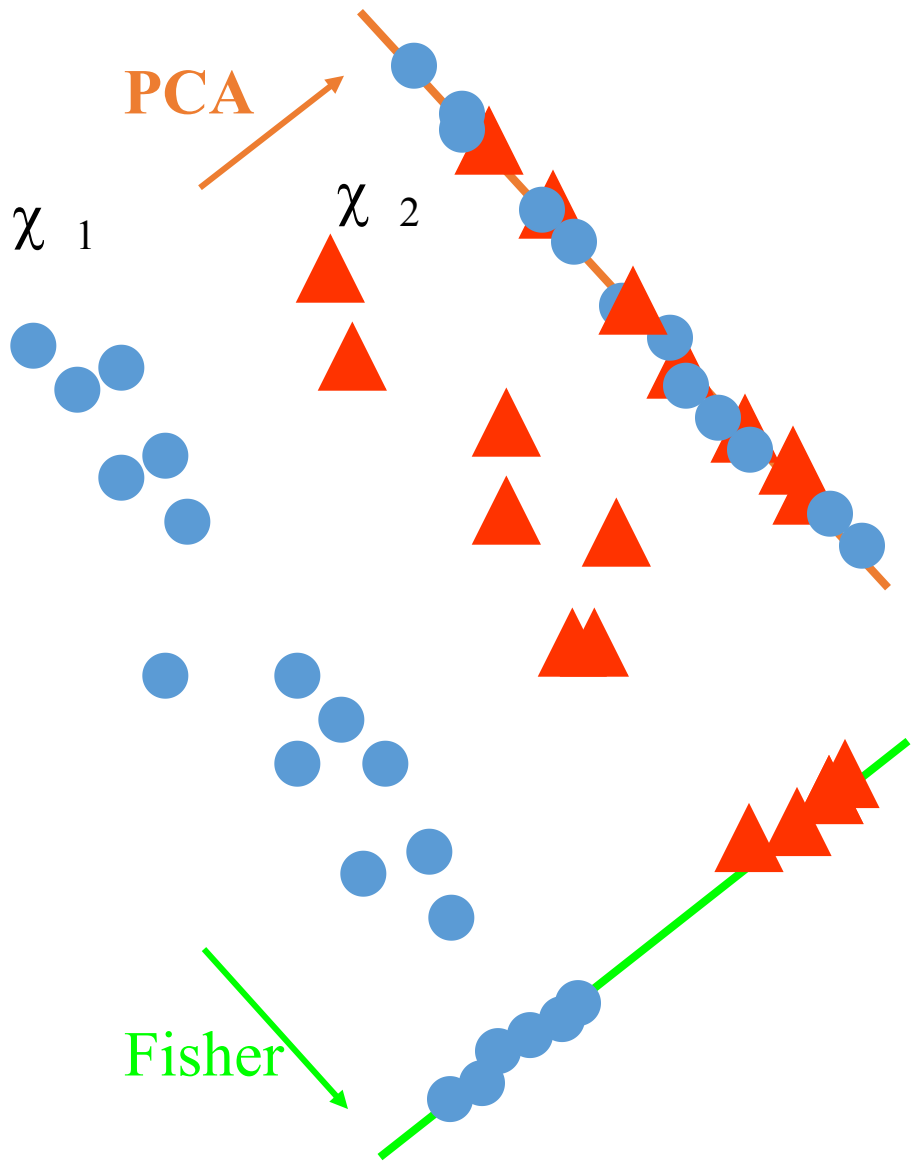
- Total scatter

$$S_T = \sum_{i=1}^{c} \sum_{x_k \in \chi_i} (x_k - \mu)(\mu_k - \mu)^T = S_B + S_W$$



- Where
  - c is the number of classes
  - μ_i is the mean of class χ_i
  - $|\chi_i|$ is number of samples of χ_i.

Eigen vs Fisher Projection



- PCA (Eigenfaces)

$$W_{PCA} = \arg\max_{W} \left| W^T S_T W \right|$$

Maximizes projected total scatter

- Fisher's Linear Discriminant

$$W_{fld} = \arg\max_{W} \frac{\left| W^T S_B W \right|}{\left| W^T S_W W \right|}$$

Maximizes ratio of projected between-class to projected within-class scatter, solved by the generalized Eigen problem:

$$S_B W = \lambda S_W W$$

A problem…

- Compute Scatter matrix in I-dimensional space
  - We need at least d data points to compute a non-singular scatter matrix.
  - E.g, d=3, we need at least 3 points, and, these points should not be co-plane (gives rank 2, instead of 3).
  - In face recognition, d is number of pixels, say 20x20=400, while number of training samples per class is small, say, n_k = 10.
  - What shall we do ?

Computing the Fisher Projection Matrix

$$W_{opt} = \arg\max_{W} \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

$$= \left[\mathbf{w}_1 \;\; \mathbf{w}_2 \;\; \ldots \;\; \mathbf{w}_m\right] \qquad (4)$$

where $\{\mathbf{w}_i \mid i = 1, 2, \ldots, m\}$ is the set of generalized eigenvectors of S_B and S_W corresponding to the m largest generalized eigenvalues $\{\lambda_i \mid i = 1, 2, \ldots, m\}$, i.e.,

$$S_B \mathbf{w}_i = \lambda_i S_W \mathbf{w}_i, \qquad i = 1, 2, \ldots, m$$

- The w_i are orthonormal
- There are at most c-1 non-zero generalized Eigenvalues, so m <= c-1
- Rank of S_w is N-c, where N is the number of training samples (=10 in AT&T data set), and c=40, so S_w can be singular and present numerical difficulty.

Dealing with Singularity of S_w

$$W = W_{fld} W_{PCA}$$

$$W_{PCA} = \arg\max_{W} \left| W^T S_T W \right|$$

$$W_{fld} = \arg\max_{W} \frac{\left| W^T W_{PCA}^T S_B W_{PCA} W \right|}{\left| W^T W_{PCA}^T S_W W_{PCA} W \right|}$$

- Since S_W is rank N-c, project training set via PCA first to subspace spanned by first N-c principal components of the training set.
- Apply FLD to N-c dimensional subspace yielding c-1 dimensional feature space.

- Fisher's Linear Discriminant projects away the within-class variation (lighting, expressions) found in training set.
- Fisher's Linear Discriminant preserves the separability of the classes.

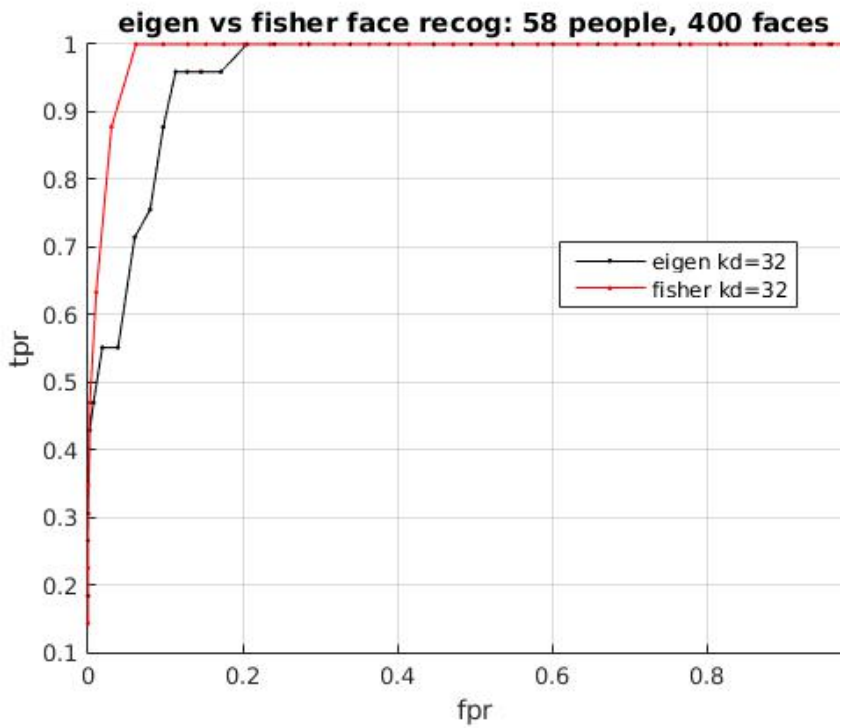
Experiment results on 417-6680 data set

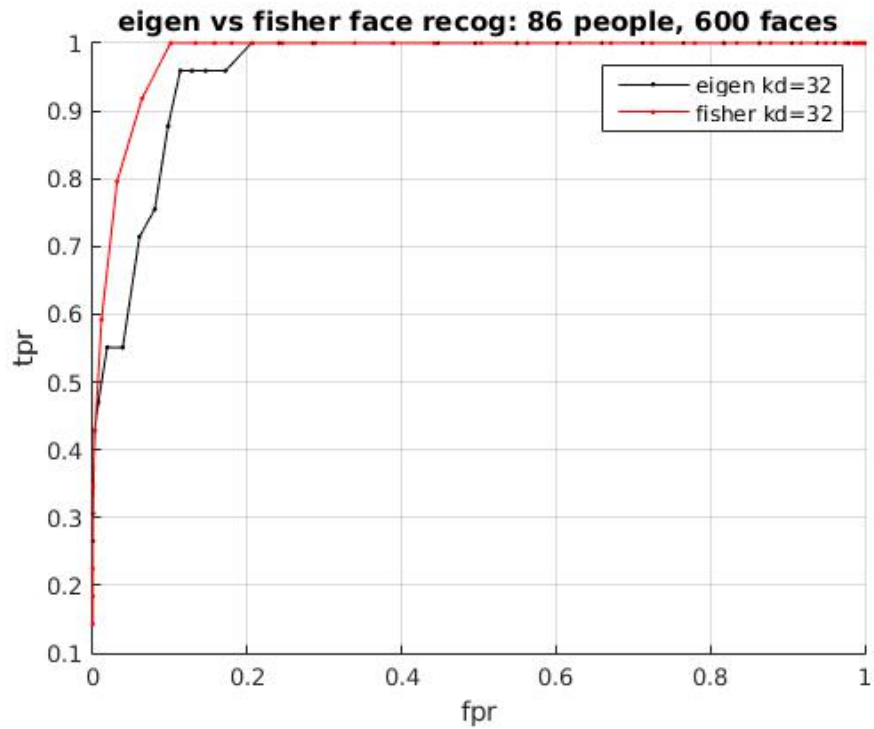
❑ Compute Eigenface model and Fisherface model

```
% Eigenface: A1
load faces-ids-n6680-m417-20x20.mat;
[A1, s, lat]=princomp(faces);

% Fisherface: A2
n_face = 600; n_subj = length(unique(ids(1:n_face)));
%eigenface kd
kd = 32; opt.Fisherface = 1;
[A2, lat]=LDA(ids(1:n_face), opt, faces(1:n_face,:)*A1(:,1:kd));
% eigenface
x1 = faces*A1(:,1:kd);
f_dist1 = pdist2(x1, x1);
% fisherface
x2 = faces*A1(:,1:kd)*A2;
f_dist2 = pdist2(x2, x2);
```

Fisher vs Eigenface performance

- Eigenface model: kd=32
- Data set: 400 faces/58 subjects, 600 faces/86 subjects





Summary

- PCA approach
  - Try to find an orthogonal rotation s.t. the first k-dimensions of the new subspace, captures most of the info of the data
  - Agnostic to the label info
  - Matlab: [A, s, eigv]=princomp(data);
- LDA approach
  - Try to find projection that maximizes the ratio of between class scatter vs within class scatter
  - Typically solved by apply PCA first to avoid within class scatter singularity
  - Utilizes label info
  - Matlab: [A, eigv]=LDA(label, opt, data);