Principle Components Analysis (PCA) and Face Recognition

Computer Vision (CS0029)

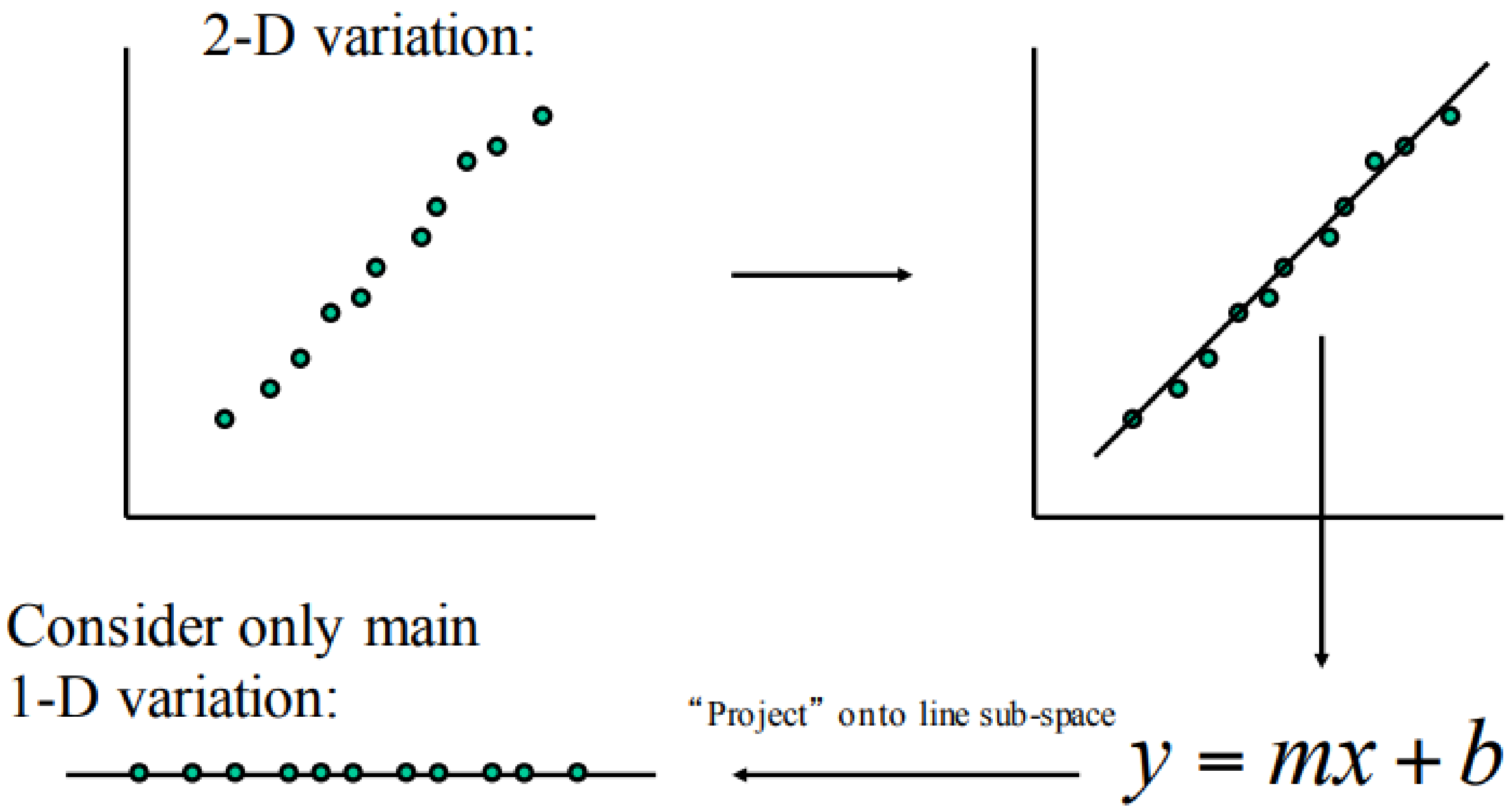
Feature Sub-Space

- A high-dimensional feature vector is often contained within a lower-dimensional “sub-space”
- When processing the feature data (for modeling or recognition), it is beneficial to deal with the lower-dimensional sub-space
  - Modeling speed, noise
- PCA offers linear approximation to the sub-space which can be reduced to only the major sub-space dimensions
  - The dimensions of the sub-space has high data variance

Outline (PCA)

- **The most basic linear algebra**
  - **Linear Basis Set**
- Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)
- Eigenvalue and eigenvector
- Connect eigenvalue and eigenvector to ellipse
  - Ellipse contour in 2D
- Extend ellipse to Gaussian
  - Covariance matrix
  - Gaussian contours
- PCA face recognition

Dimensionality Reduction



Linear Basis Set

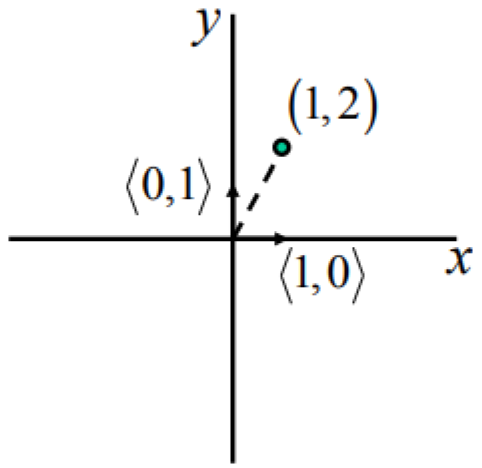
- 2-D basis set
  - Coordinate is weights of bases

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

x,y coordinates

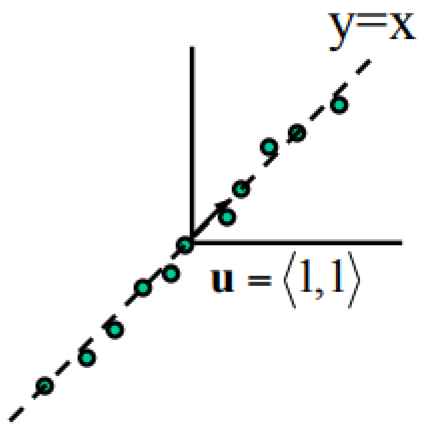
x-axis

y-axis



Linear Basis Set

- 1-D sub-space basis set



2-D space:

$$\mathbf{x_i} = a_i \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b_i \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

1-D sub-space:

$$y_i = \gamma_i \cdot \mathbf{u} = \gamma_i \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Linear Basis Set

- Generate smaller dimension basis for feature vectors

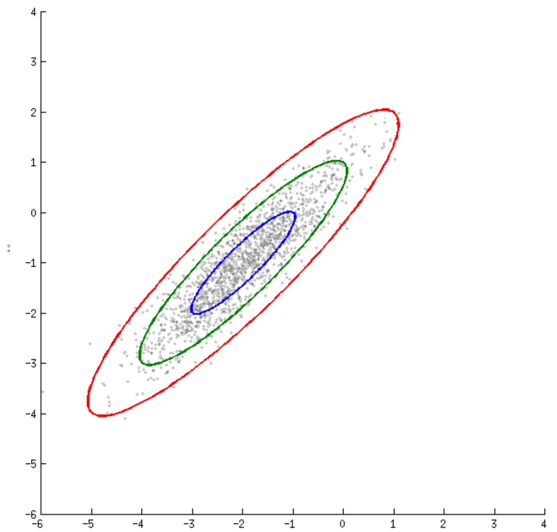
  - $x_i = \gamma_1 u_1 + \gamma_2 u_2 + \gamma_3 u_3 + \ldots + \gamma_m u_m$

    where $\dim(x) > m$

Outline (PCA)

- The most basic linear algebra
  - Linear Basis Set
- **Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)**
- Eigenvalue and eigenvector
- Connect eigenvalue and eigenvector to ellipse
  - Ellipse contour in 2D
- Extend ellipse to Gaussian
  - Covariance matrix
  - Gaussian contours
- PCA face recognition

Principal Components Analysis (PCA)



- The main idea for PCA
  - Fit a multi-dimensional Gaussian around data
    - Assumption: data distribution is close to a Gaussian distribution
    - Use covariance of data to model Gaussian
- Select only those dimensions capturing most of the variance in data
  - Reduce dimensionality

Outline (PCA)

- The most basic linear algebra
  - Linear Basis Set
- Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)
- **Eigenvalue and eigenvector**
- Connect eigenvalue and eigenvector to ellipse
  - Ellipse contour in 2D
- Extend ellipse to Gaussian
  - Covariance matrix
  - Gaussian contours
- PCA face recognition

PCA Primer: Equation for Eigenvalues

Stretch/Scale $\mathbf{x}$
(Eigenvalue)

Square matrix

$$\mathbf{Ax} = \underline{\lambda \mathbf{x}}$$

Same direction as $\mathbf{Ax}$

Certain vector
(Eigenvector)

$$\mathbf{Ax} - \lambda\mathbf{x} = \mathbf{0}$$

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

$\mathbf{A}$ is required to be square matrix for PCA

$$\boldsymbol{K} = \boldsymbol{A} - \boldsymbol{\lambda I}$$
$$\boldsymbol{Kx} = \mathbf{0}$$

If x is not zero vector,
K is singular matrix,
and det(K)=0

PCA Primer: Equation for Eigenvalues

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$$

$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 1-\lambda & 2 \\ 2 & 4-\lambda \end{bmatrix}$$

$$\det(\mathbf{A} - \lambda\mathbf{I}) = (1-\lambda)(4-\lambda) - (2)(2) = \lambda^2 - 5\lambda = 0$$

$$\text{two roots: } \lambda_1 = 0, \ \lambda_2 = 5$$

PCA Primer: Equation for Eigenvalues

$$(\mathbf{A} - \lambda_i \mathbf{I})\mathbf{x}_i = \mathbf{0}$$

$$(\mathbf{A} - \lambda_1 \mathbf{I})\mathbf{e}_1 = \mathbf{0}$$

$$\left(\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right) \mathbf{e}_1 = \mathbf{0}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{e}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$(\mathbf{A} - \lambda_2 \mathbf{I})\mathbf{e}_2 = \mathbf{0}$$

$$\left(\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} - \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \right) \mathbf{e}_2 = \mathbf{0}$$

$$\begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \mathbf{0}$$

$$\mathbf{e}_2 = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

PCA Primer: Test Our Results

$$\mathbf{A}\mathbf{e}_1 \stackrel{?}{=} \lambda_1 \mathbf{e}_1$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} (1 \cdot 2 - 2 \cdot 1) \\ (2 \cdot 2 - 4 \cdot 1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = 0 \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\mathbf{A}\mathbf{e}_2 \stackrel{?}{=} \lambda_2 \mathbf{e}_2$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} (1 \cdot 1 + 2 \cdot 2) \\ (2 \cdot 1 + 4 \cdot 2) \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \end{bmatrix} = 5 \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Practice

- $A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}$, calculate eigen values of A?
  - $Ax = \lambda x \;\; \Rightarrow \;\; Ax - \lambda x = 0 \;\Rightarrow (A - \lambda I)x = 0$
  - $(A - \lambda I)$ is also a 2x2 matrix and $\det(A - \lambda I) = 0$ if eigenvector x is not 0 vector

- Eigenvector $(x_1, and x_2)$
  - Solve $Ax_1 = \lambda_1 x_1$ and $Ax_2 = \lambda_2 x_2$

PCA Primer: Symmetric Matrices

- If A is symmetric ($A = A^T$)
  - Real-valued eigenvalues
  - Eigenvectors can be chosen orthonormal
  - Can be factorized as
    - $A = Q\Lambda Q^T$
      where Q orthonormal ($Q^T Q = I$) and Λ is diagonal
  - Eigenvalues go into diagonal entries of Λ
    - Convention: put largest eigenvalues first (descending values)
  - Corresponding orthogonal eigenvectors are normalized (to become orthonormal) and go into columns of Q

Eigen Decomposition: Numpy

```python
import numpy as np

A = np.array([[5, 1], [3, 3]])
eValue, eVector = np.linalg.eig(A) #decompose matrix A
print( "A=\n", A )
print( "\nEigenValues=\n", eValue )
print( "\nEigenVectors(Columns)=\n", eVector )

eigenIndex = 0 #the first eigen value and vector
print("\nEigenIndex: ", eigenIndex)
v = eVector[:,eigenIndex].reshape(2,1) #eigen vector
print("\nv=\n", v)
Av = A.dot(v) #A*v
vx = v*eValue[eigenIndex] #v*x
print("\nAv=\n", Av) #compare A*v = v*x
print("\nvx=\n", vx)

eigenIndex = 1
print("\n\nEigenIndex: ", eigenIndex)
v = eVector[:,eigenIndex].reshape(2,1)
print("\nv=\n", v)
Av = A.dot(v)
vx = v*eValue[eigenIndex]
print("\nAv=\n", Av)
print("\nvx=\n", vx)
```

```
A=
 [[5 1]
 [3 3]]

EigenValues=
 [6. 2.]

EigenVectors(Columns)=
 [[ 0.70710678 -0.31622777]
 [ 0.70710678  0.9486833 ]]

EigenIndex:  0

v=
 [[0.70710678]
 [0.70710678]]

Av=
 [[4.24264069]
 [4.24264069]]

vx=
 [[4.24264069]
 [4.24264069]]


EigenIndex:  1

v=
 [[-0.31622777]
 [ 0.9486833 ]]

Av=
 [[-0.63245553]
 [ 1.8973666 ]]

vx=
 [[-0.63245553]
 [ 1.8973666 ]]
```

Outline (PCA)

- The most basic linear algebra
  - Linear Basis Set
- Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)
- Eigenvalue and eigenvector
- **Connect eigenvalue and eigenvector to ellipse**
  - **Ellipse contour in 2D**
- Extend ellipse to Gaussian
  - Covariance matrix
  - Gaussian contours
- PCA face recognition

Positive Definite Matrices

- Matrix A is positive definite for all non-zero x if
  - Symmetric and $x^T A x > 0$

Positive semi-definite if

$$\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & b \\ b & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} > 0$$

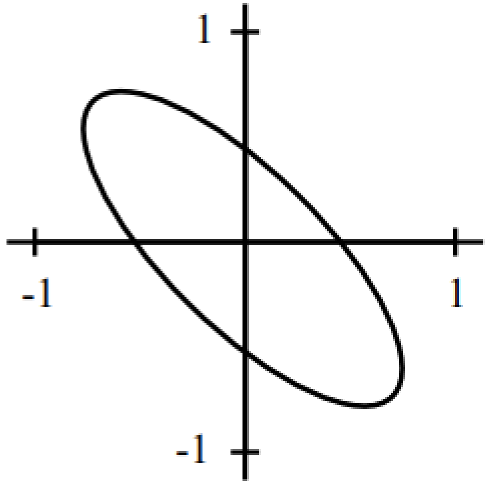
$$ax^2 + 2bxy + cy^2 > 0$$

- Recall equation for ellipse
  - Ellipse: $ax^2 + 2bxy + cy^2 \ = 1$
  - Center at (0,0)

Ellipse Factorization

- Consider the rotated ellipse:
- Know the shape of ellipse?
  - Calculate axes and length

$$5x^2 + 8xy + 5y^2 = 1$$

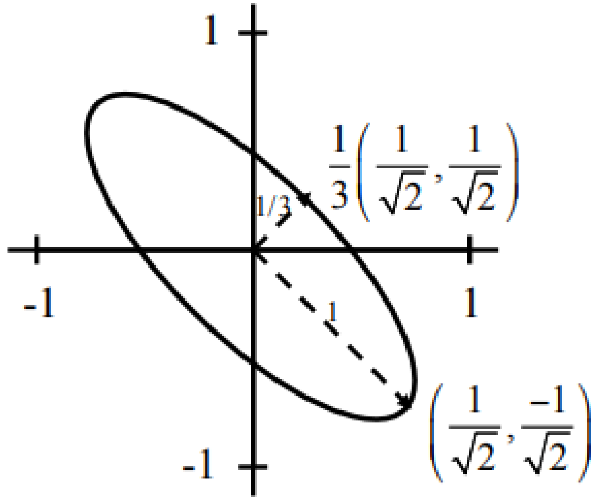


Ellipse Factorization

$$5x^2 + 8xy + 5y^2 = 1 \quad \rightarrow \quad a = 5,\ b = 4,\ c = 5$$

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = 1$$

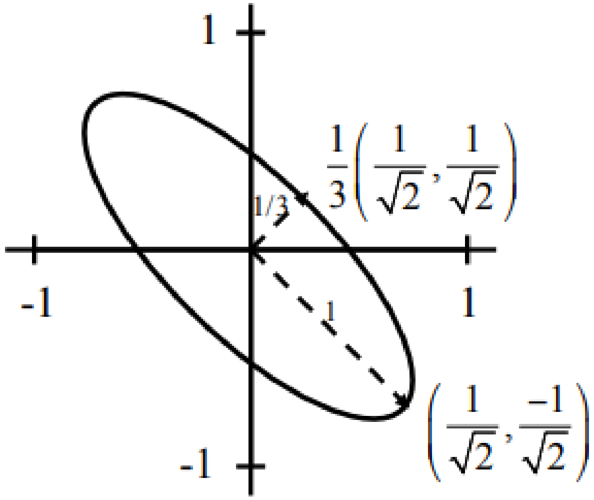
$$\mathbf{A} = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix} = \mathbf{Q}\Lambda\mathbf{Q}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}} \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}_{\sqrt{2}}$$



Axes of ellipse point along eigenvectors

Half-lengths of axes are $1/\sqrt{\lambda_i}$ (NOTE: bigger eigenvalues give shorter axes!)

Eigenvector projection



- The matrix Q^T acts as a rotation matrix

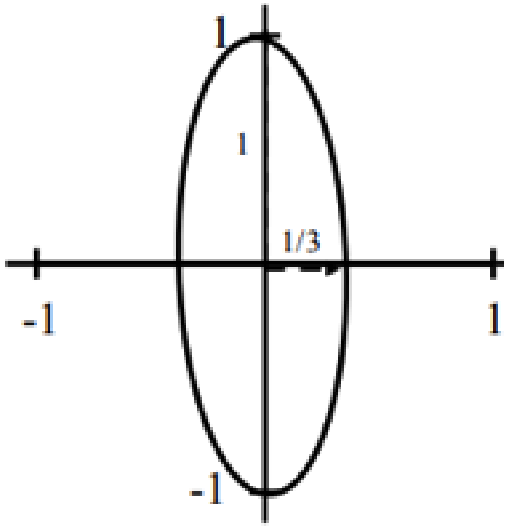
$$\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}=\mathbf{x}^{\mathrm{T}}\left(\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathrm{T}}\right)\mathbf{x}=\left(\mathbf{x}^{\mathrm{T}}\mathbf{Q}\right)\mathbf{\Lambda}\left(\mathbf{Q}^{\mathrm{T}}\mathbf{x}\right)=\mathbf{X}^{\mathrm{T}}\mathbf{\Lambda}\mathbf{X}$$

$$\mathbf{X}=\mathbf{Q}^{\mathrm{T}}\mathbf{x}=\begin{bmatrix}\hat{\mathbf{e}}_1^T\mathbf{x}\\ \hat{\mathbf{e}}_2^T\mathbf{x}\end{bmatrix}$$

Rotate previous axes:

$$\mathbf{X}_1=\mathbf{Q}^{\mathrm{T}}\mathbf{x}_1=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}_{\sqrt{2}}\begin{bmatrix}0\\ 1\end{bmatrix}=\begin{bmatrix}1\\ -1\end{bmatrix}_{\sqrt{2}}$$

$$\mathbf{X}_2=\mathbf{Q}^{\mathrm{T}}\mathbf{x}_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}_{\sqrt{2}}\begin{bmatrix}1\\ 0\end{bmatrix}=\begin{bmatrix}1\\ 1\end{bmatrix}_{\sqrt{2}}$$



Outline (PCA)

- The most basic linear algebra
  - Linear Basis Set
- Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)
- Eigenvalue and eigenvector
- Connect eigenvalue and eigenvector to ellipse
  - Ellipse contour in 2D
- **Extend ellipse to Gaussian**
  - **Covariance matrix**
  - **Gaussian contours**
- PCA face recognition

Gaussian Density

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{K}|^{\frac{1}{2}}} \cdot e^{\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \mathbf{K}^{-1}(\mathbf{x}-\mathbf{m})\right]}$$

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Mahalanobis distance: $(x-m)^T K^{-1}(x-m) = C$

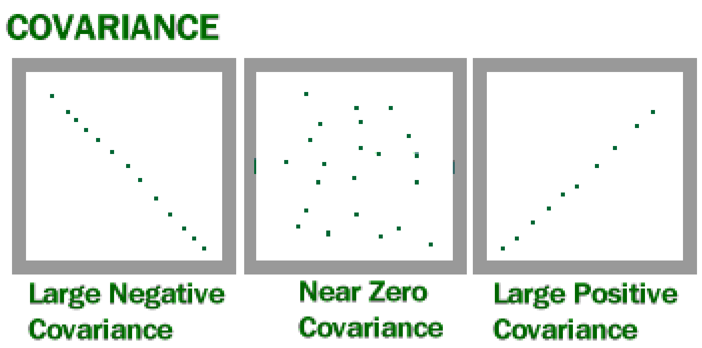
Focus of all points at given distance C from mean

(i.e., variance contour with $C = \#stdev^2$)

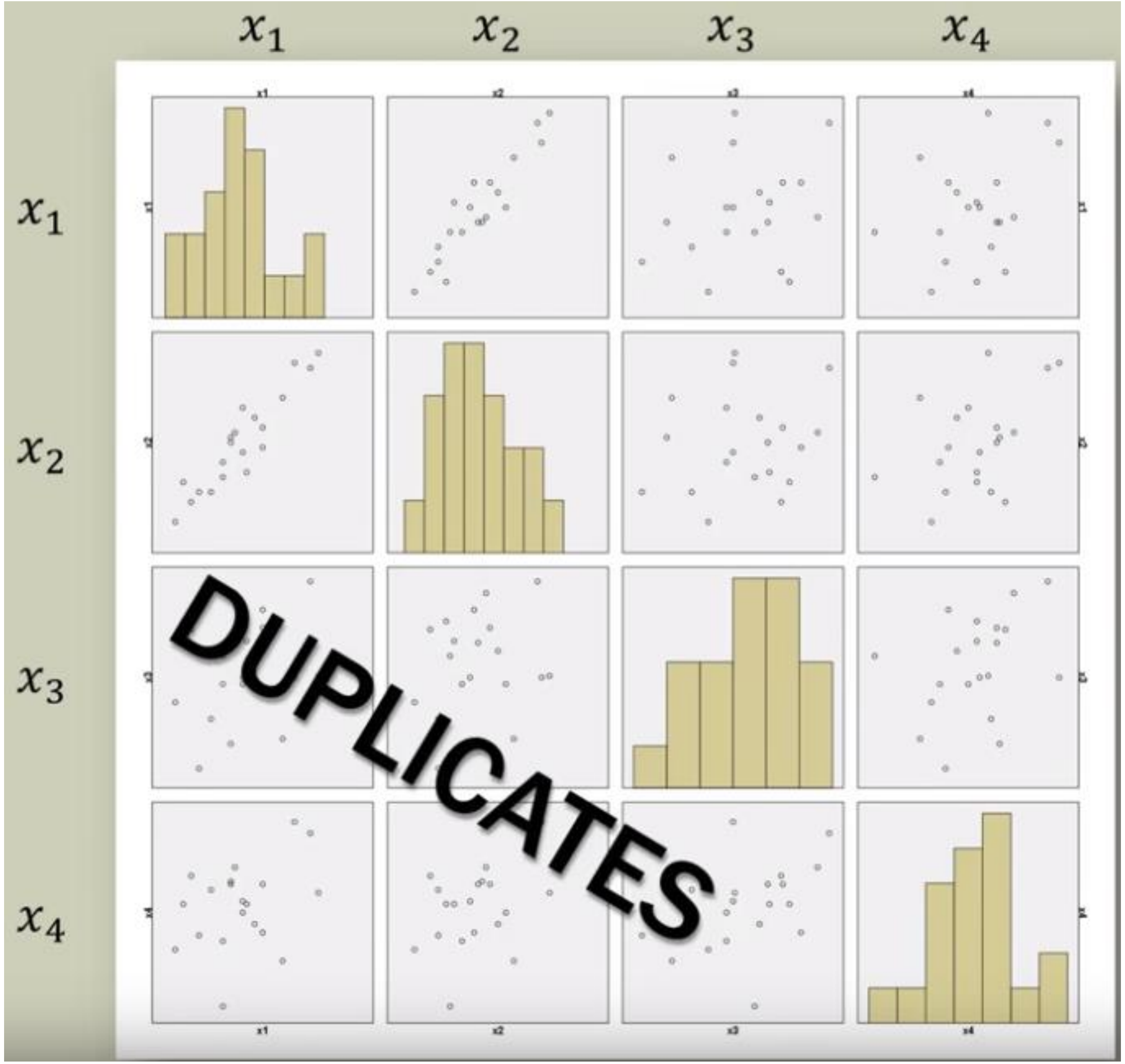
Covariance

- Covariance is one of the family of statistical measurement used to analyze the linear relationship between two variables.

- Positive covariance: increasing linear relation
- Negative covariance: decreasing linear relation
  - Not strength



- Covariance matrix: covariances from multiple varialbes
  - symmetric


x_1 x_2 x_3 x_4
x_1
x_2
x_3
x_4
DUPLICATES

Four Variables Covariance Matrix

	x_1	x_2	x_3	x_4
x_1	$Var(x_1)$	$Cov(x_1, x_2)$	$Cov(x_1, x_3)$	$Cov(x_1, x_4)$
x_2		$Var(x_2)$	$Cov(x_2, x_3)$	$Cov(x_2, x_4)$
x_3			$Var(x_3)$	$Cov(x_3, x_4)$
x_4				$Var(x_4)$

Variance: $\mathrm{Var}(X) = \mathrm{E}\left[(X-\mu)^2\right]$

Covariance Matrix Equation

Covariance matrix (K) of X which is collection of data points:

K is symmetric (and positive [semi-]definite)

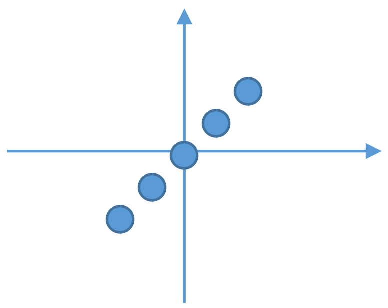
$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$$

$$\mathrm{K}_{X_i X_j} = \mathrm{cov}[X_i, X_j] = \mathrm{E}[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])]$$

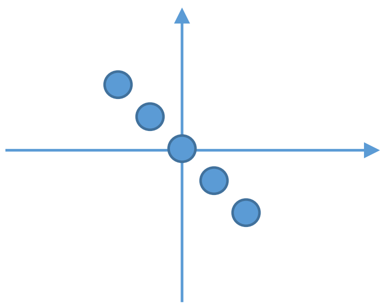
$$\mathrm{K}_{\mathbf{XX}} = \begin{bmatrix} \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_1 - \mathrm{E}[X_1])(X_n - \mathrm{E}[X_n])] \\ \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_2 - \mathrm{E}[X_2])(X_n - \mathrm{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_1 - \mathrm{E}[X_1])] & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_2 - \mathrm{E}[X_2])] & \cdots & \mathrm{E}[(X_n - \mathrm{E}[X_n])(X_n - \mathrm{E}[X_n])] \end{bmatrix}$$

Example

- Two variables (x, y) case, given 5 data points
  - (-2,-2), (-1,-1), (0,0), (1,1), (2,2)
  - $\begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$



- Two variables (x, y) case, given 5 data points
  - (-2,2), (-1,1), (0,0), (1,-1), (2,-2)
  - $\begin{bmatrix} 5 & -5 \\ -5 & 5 \end{bmatrix}$



3 Variables Covariance Matrix

- 5 data samples
  - [-2, 1, 2]
  - [-1, 3, 1]
  - [0, 5, 0]
  - [1, 7, -1]
  - [2, 9, -2]

Variance: $\operatorname{Var}(X) = \operatorname{E}\left[(X-\mu)^2\right]$

$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$$

$$\operatorname{K}_{X_i X_j} = \operatorname{cov}[X_i, X_j] = \operatorname{E}[(X_i - \operatorname{E}[X_i])(X_j - \operatorname{E}[X_j])]$$

$$\operatorname{K}_{\mathbf{X}\mathbf{X}} = \begin{bmatrix} \operatorname{E}[(X_1 - \operatorname{E}[X_1])(X_1 - \operatorname{E}[X_1])] & \operatorname{E}[(X_1 - \operatorname{E}[X_1])(X_2 - \operatorname{E}[X_2])] & \cdots & \operatorname{E}[(X_1 - \operatorname{E}[X_1])(X_n - \operatorname{E}[X_n])] \\ \operatorname{E}[(X_2 - \operatorname{E}[X_2])(X_1 - \operatorname{E}[X_1])] & \operatorname{E}[(X_2 - \operatorname{E}[X_2])(X_2 - \operatorname{E}[X_2])] & \cdots & \operatorname{E}[(X_2 - \operatorname{E}[X_2])(X_n - \operatorname{E}[X_n])] \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{E}[(X_n - \operatorname{E}[X_n])(X_1 - \operatorname{E}[X_1])] & \operatorname{E}[(X_n - \operatorname{E}[X_n])(X_2 - \operatorname{E}[X_2])] & \cdots & \operatorname{E}[(X_n - \operatorname{E}[X_n])(X_n - \operatorname{E}[X_n])] \end{bmatrix}$$

3 Variables Covariance Matrix

- 5 data samples
  - [-2, 1, 2]
  - [-1, 3, 1]
  - [0, 5, 0]
  - [1, 7, -1]
  - [2, 9, -2]

- Covariance matrix: $\begin{bmatrix} 5 & 4 & -5 \\ 4 & 8 & -4 \\ -5 & 4 & -5 \end{bmatrix}$

- 5 data samples (3 variable x, y, z)
  - $\begin{array}{ccc} x & y & z \end{array}$
  - A = $\begin{bmatrix} 2 & 1 & 2 \\ -1 & 3 & 1 \\ 0 & 5 & 0 \\ 1 & 7 & -1 \\ 2 & 9 & -2 \end{bmatrix}$
  - B = $\begin{bmatrix} 2-\mu_x & 1-\mu_y & 2-\mu_z \\ -1-\mu_x & 3-\mu_y & 1-\mu_z \\ 0-\mu_x & 5-\mu_y & 0-\mu_z \\ 1-\mu_x & 7-\mu_y & -1-\mu_z \\ 2-\mu_x & 9-\mu_y & -2-\mu_z \end{bmatrix}$

- Covariance matrix: $\frac{(B^T B)}{N}$
  - N is number of samples

Plotting Gaussian Density Function Contours (Physical Meaning of Covariance Matrix)

- K is covariance matrix
  - $K = \frac{(B^T B)}{N}$ (check the previous page)
- Describes a (rotated) ellipse (at a C variance contour, or squared stdev contour) centered around mean:

$$(x - m)^T K^{-1} (x - m) = C$$

- Thus we can factorize K^{-1} into eigenvectors (axes) and eigenvalues to give direction and half-lengths of ellipse axes

$$K = Q\Lambda Q^T$$
$$K^{-1} = Q\Lambda^{-1} Q^T$$

Axes

- To compute the ellipse at Gaussian “variance” contour C $(= \#stdev^2)$:
  - From matrix K^{-1} (inverse covariance)
    - Axes are columns in Q,
    - Half-lengths of axes are $\frac{\sqrt{C}}{\sqrt{\lambda_i}}$, with λ_i from Λ^{-1}
  - From matrix K (covariance matrix)
    - Axes are columns in Q
    - Half-lengths of axes are $\sqrt{C\lambda_i}$, with λ_i from Λ

Putting it all together

Ellipse formula:

$$x^T A x = 1$$

Ellipse defined by eigenvectors/values of *A*

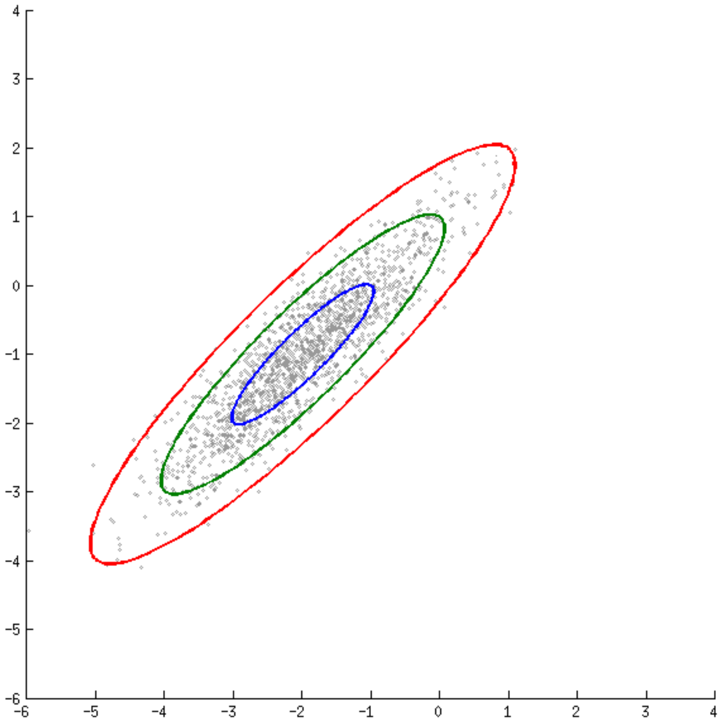
Gaussian & Mahalanobis:

$$(x - m)^T K^{-1} (x - m) = C$$

$$A = K^{-1}$$

Ellipse defined by eigenvectors/values of K^{-1}

Or use K and just “flip” the eigenvalues (saves doing the inverse operation)

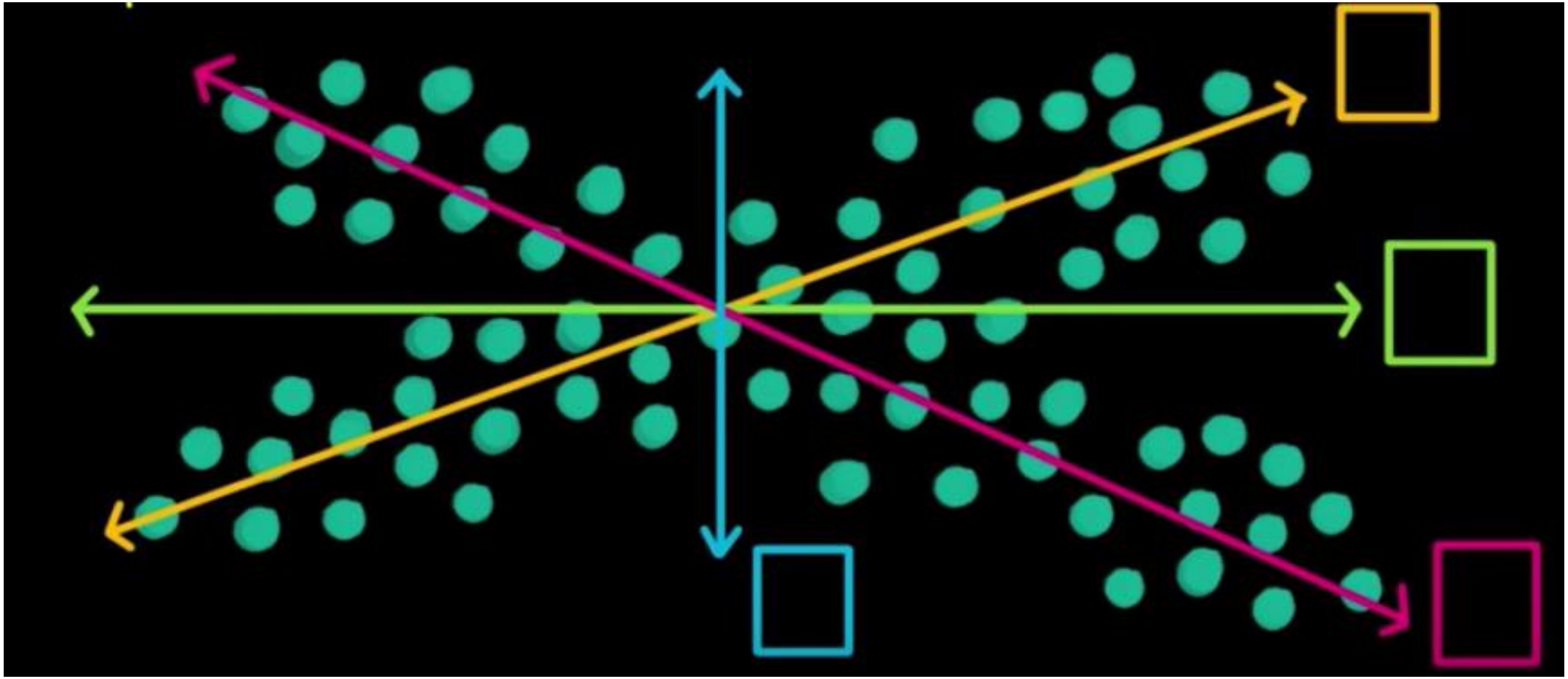


<u>Larger</u> values of *C* give <u>bigger</u> variance contour ellipses

PCA

- Reduce dimensionality
  - Use a Gaussian (covariance matrix) to model a collection of data samples
    - Assumption: data distribution is similar to Gaussian
  - Eigen decomposition to get eigen vector and eigen values
  - Sort eigen vectors by eigen values
  - Use n eigen vectors with largest n eigen values to create a sub-space
  - Project data samples to this sub-space ← dimension reduction
    - x=[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, ………. a_m,]
    - x^{sub}= $b_1\ e_1 + b_2\ e_2 + b_3\ e_3 + \cdots b_n\ e_n$
      - new coordinate in the new subspace is [b_n , b_n , b_n , … , b_n]
      - n < m

The Largest Two Principle Component



Outline (PCA)

- The most basic linear algebra
  - Linear Basis Set
- Connect PCA to Gaussian Distribution (Ellipse shape distribution in 2D)
- Eigenvalue and eigenvector
- Connect eigenvalue and eigenvector to ellipse
  - Ellipse contour in 2D
- Extend ellipse to Gaussian
  - Covariance matrix
  - Gaussian contours
- **PCA face recognition**

Face Recognition

- Old method

- Project face images to Gaussian feature space spanning significant variation among known face image

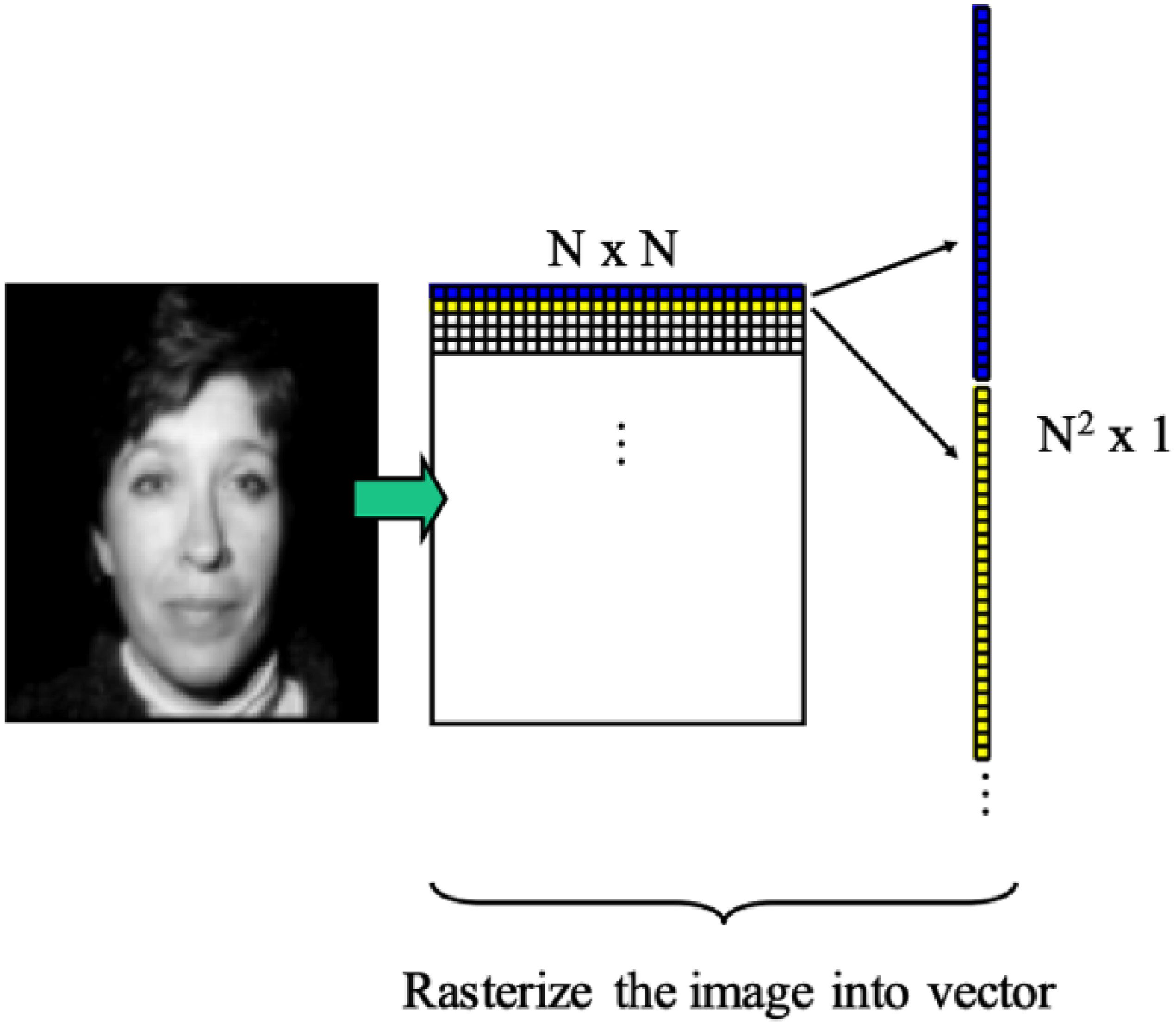
- Significant features in eigenspace for projection are call “EigenFace”
    - The eigenvectors capturing the majority of “variance” of data
        - Largest vector correspond to the largest component variance
    - Project face image (vector) onto selected top eigenvectors

Face Recognition

- Recognition achieved by comparing weights/coefficients of new face (after projection onto eigenspace) to other stored face weights/coefficients
  - Distance calculation more compact and efficient (uses small number of weights/ coefficients)

- Calculate distance from face space or individual
  - Does it look like a face?
  - Does it look like “Casey”?

Input Data



Input Data

- Compute mean face image Ψ
  - From set of rasterized face images Υ_i (training image)

- Subtract mean from images
  - Remove the mean using $\Phi_i = \Upsilon_i$- Ψ
  - From matrix of training faces
    - A = $[\Phi_0\ \Phi_1 \Phi_2\ \Phi_3\ \ldots\ \Phi_{M-1}]$
    - Matrix size: N^2*M

- Compute covariance matrix of A

Eigen Trick

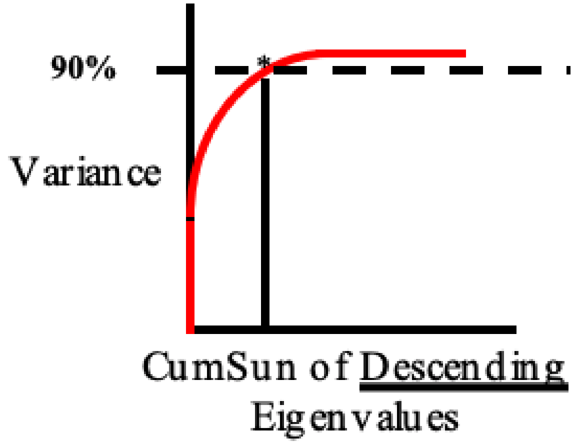
- Compute eigenvectors and eigenvalues of A from its covariance matrix
  - Gaussian sub-space
- Number of face image(M) is much less than the dimension of space (N^2)
- Thus only M-1 meaningful eigenvectors in the matrix A
  - Remaining eigenvectors have eigenvalues = 0
- Can solve using a much smaller M*M matrix and convert back to N^2*1

Eigen Trick

- Consider the eigenvectors v_i of $A^T A$
  - Recall AA^T is how to compute covariance matrix K for Φ_i data
    - ($A^T A$) v_i =$\lambda_i v_i$
  - Pre-multiplying both sized by A, we get
    - $A(A^T A)$ v_i =$\lambda_i A v_i$
  - Thus $u_i = Av_i$ are eigenvectors of AA^T
    - Covariance matrix : (AA^T)[Av_i] = $\lambda_i[Av_i]$
  - Size comparison
    - Matrix $A^T A$ is size M*M
    - Matrix AA^T (a covariance) is size N^2* N^2
  - Hence, compute eigenvectors/eigenvalues from $A^T A$ and use $u_i = Av_i$ to recover the desired dimensionality
    - Make sure to normalize the u_i to make them unit vectors

Compute Eigenvectors

- Retain only top m eigenvectors (u_k)
  - Determine from strength of eigenvalues
  - These eigenvectors are the “EigenFaces”
- Accumulate eigenvalues until reach desired percentage of total sum of eigenvalues
  - Pre-sort eigenvalues from largest to smallest
  - Considered as “% variance captured”
  - Typically use around 90%



Projection into Face Space

- Project each rasterized (mean-subtracted) face image into sub-space (m eigenvectors)
  - Project onto each eigenvector (“EigenFace”)
    - $\omega_T = u_k^T \,.\, \Phi_i$

- Keep projection coefficients as representation of rasterized image
  - $\Phi_i \rightarrow \Omega_i = [\omega_1 \omega_2 \omega_3 \; \ldots \omega_m]^T$

Reconstruction from Face Space

- Reconstruct face image from sub-space
  - Reconstruct from projection coefficients on each eigenvector
  - $\Phi_{recon} = \sum \omega_k u_k$

- Add back mean face
  - $\Upsilon_{recon} = \Phi_{recon} + \Psi$

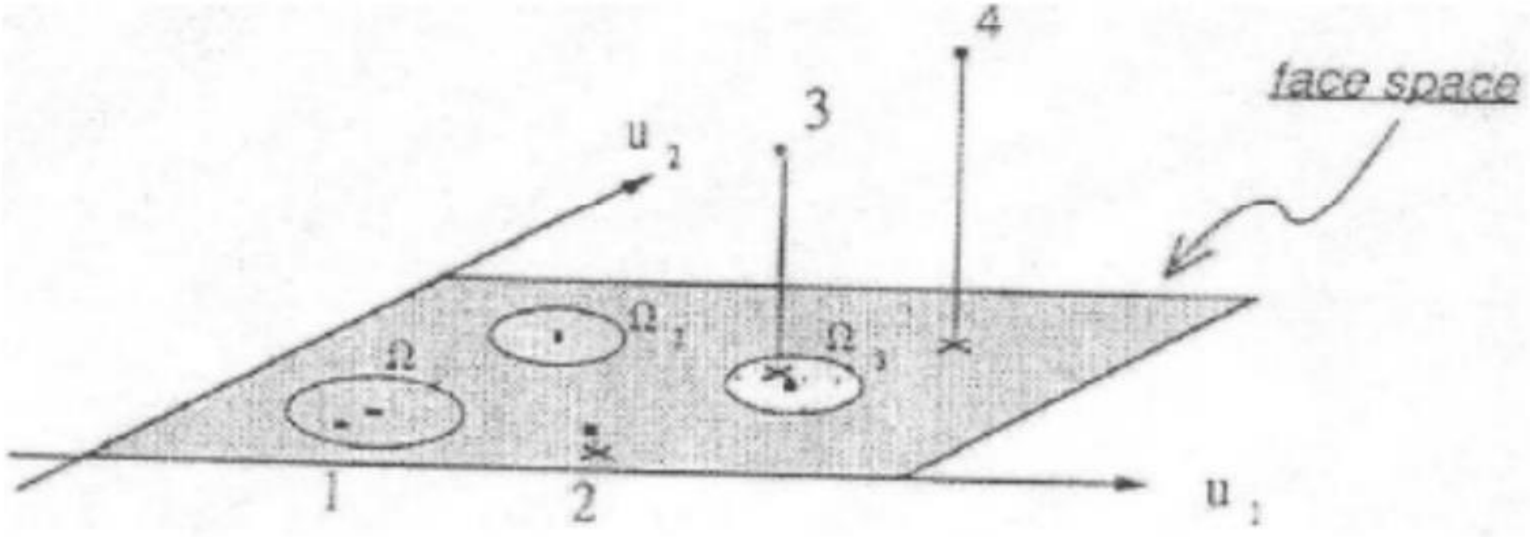
Recognition Pipeline

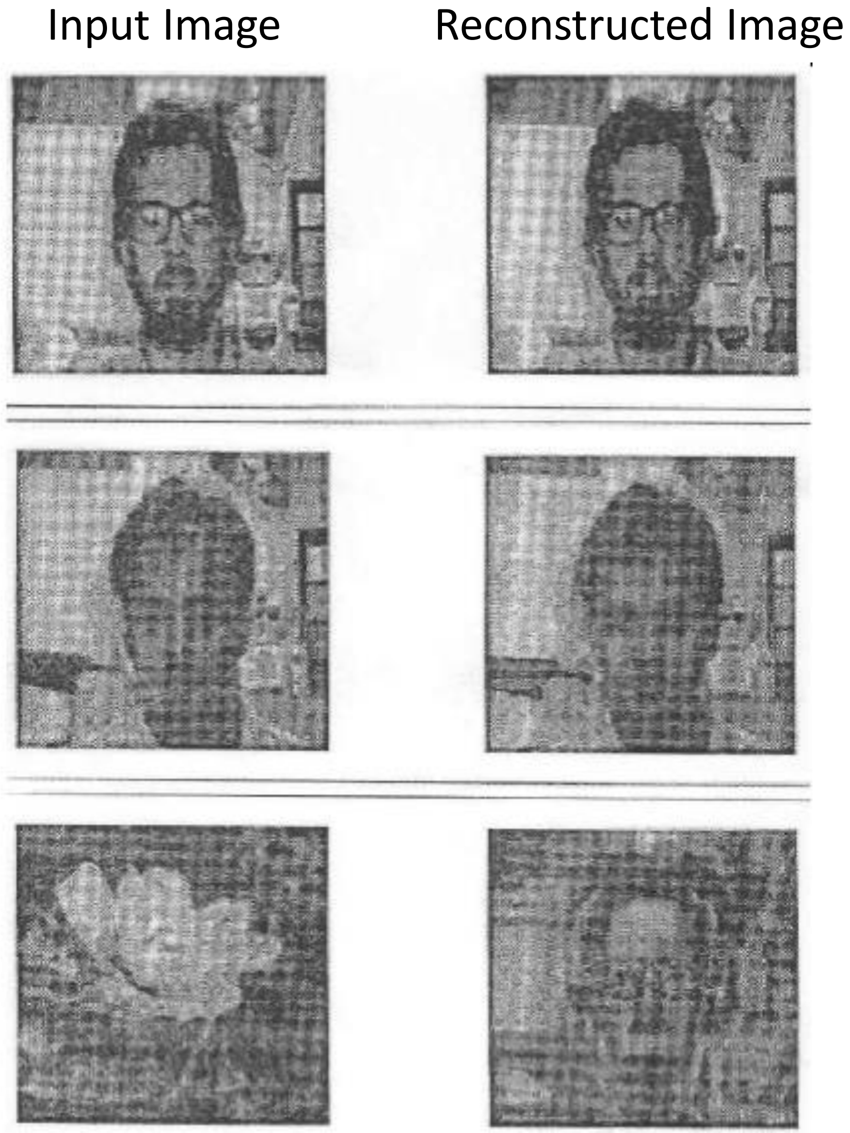
- Get new face image Φ
  - Rasterize
  - Mean-subtract using Ψ
- Compute Ω
- Use Sum-of-Squared-Error(SSE) to other faces in the database
  - Find best match for database items Ω_i (assumes that new face is in the database)
  - ID = $argmin_i \; \|\Omega - \Omega_i\|^2$
  - If $\|\Omega - \Omega_{ID}\|^2 < Threshold_{recog}$, Ω is ID

Face Detection

- Get new image Φ
  - Rasterize
  - Mean-subtract using Ψ
- Compute Ω
- Reconstruct image Φ_{recon} from Ω
- Compute $\|\Phi - \Phi_{recon}\|^2$
- If $\|\Phi - \Phi_{recon}\|^2 < Threshold_{detect}$, Φ is face

Face Detection





Principal component (eigenvector) u_k

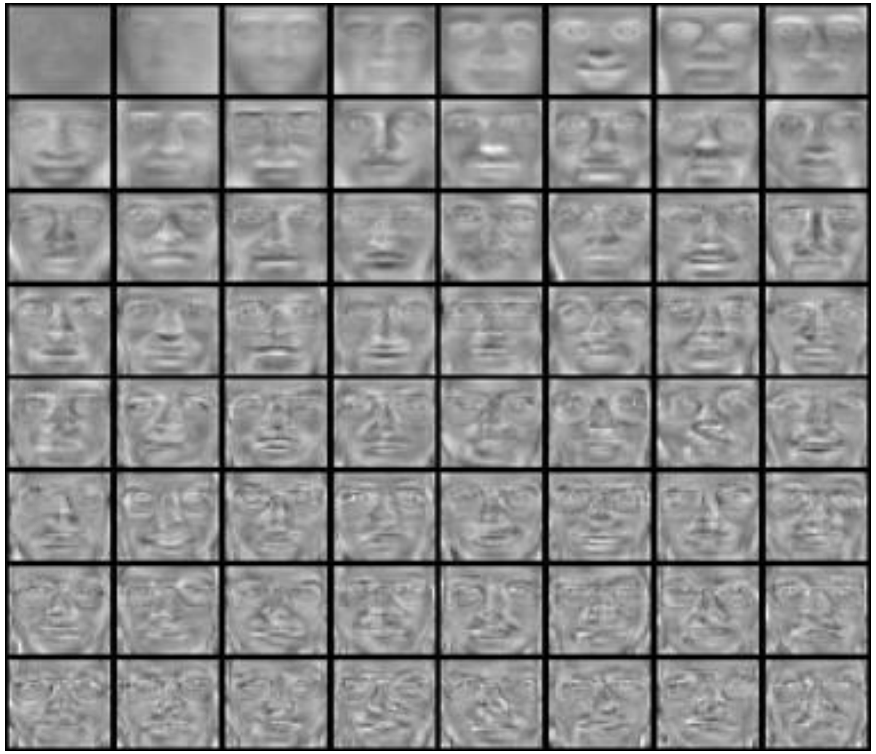
$\mu + 3\sigma_k u_k$

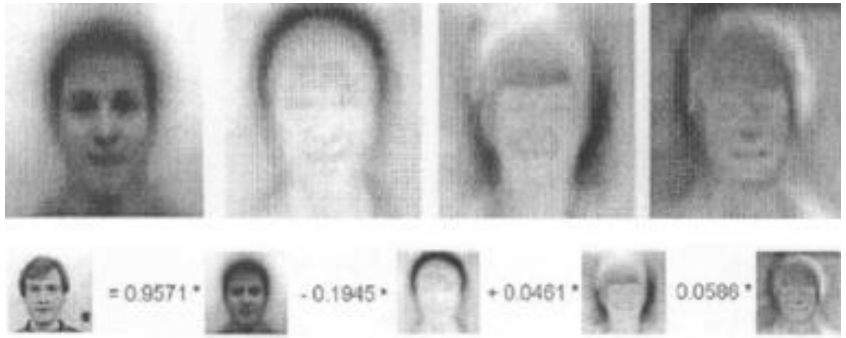
$\mu - 3\sigma_k u_k$

Training dataset

Mean face Ψ

Eigen Faces





Face reconstruction

Query image

Found image

Limitation

- Background variation and misalignment