# Dimension Reduction

## Principle Component Analysis
## Linear Discriminant Analysis

Hosoo Lee

Curse of dimensionality

Linear Algebra Background

Dimensionality Reduction

- PCA (Principle Component Analysis)
- LDA (Linear Discriminant Analysis)

Application of PCA and LDA - Face Recognition

# Curse of dimensionality

# Curse of Dimensionality

**We, as human, are experts in 1D, 2D, 3D, a bit less in 4D(time) and less so afterwards**

**In high dimensions (eg 20 is enough), counter intuitive properties appear**

**Eg:**

- **Sparsity**
- **Concentration of distances**

**…**

**Which we try to model here, to better understand why things go wrong (or not as good)**

# High dimensionality

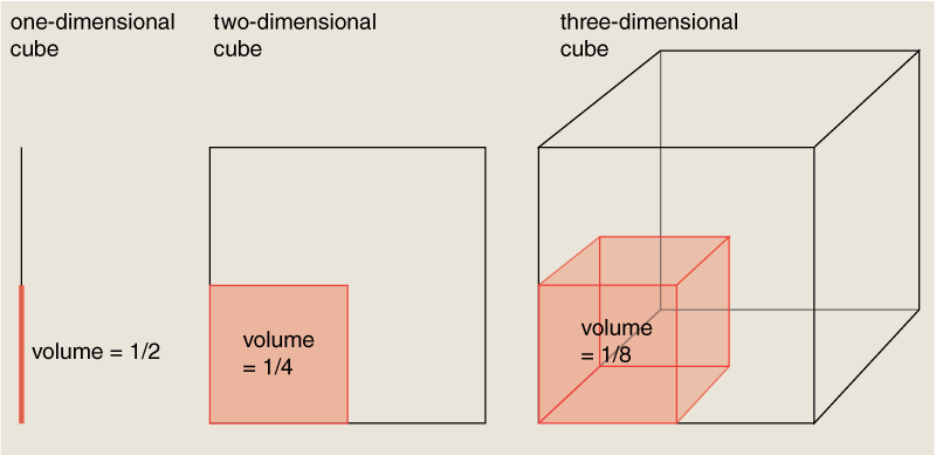
Imagine a data sample in $[a, b]^M$
We quantify every dimension with $k$ bins

To estimate the distribution
we require $n$ sample in each bin in average

- $M = 1$ : $N \sim nk$
- $M = 2$ : $N \sim nk^2$
- $M = 3$ : $N \sim nk^3$

$\vdots$

- $M$ : $N \sim nk^M$




Ex) $k = 10, n = 10, M = 6$
$\Rightarrow$ $N \sim 10,000,000$ samples required

# Curse of Dimensionality

## Sparsity

- $N$ samples
- $M$ dimensions
- $k$ quantization steps

$\Rightarrow$ $n$ samples per bin

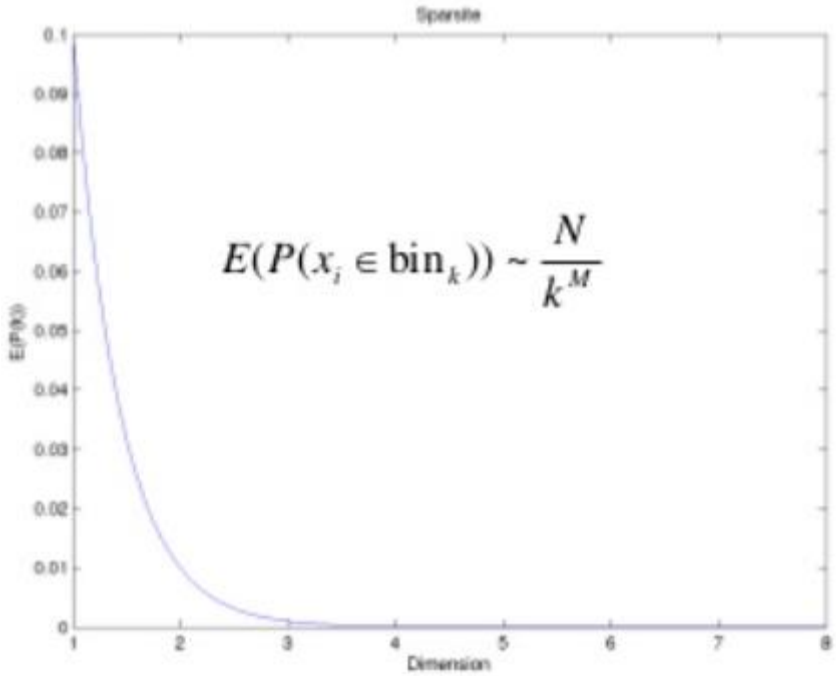
$$n \sim \frac{N}{k^M}$$

or

$$N \sim k^M$$

to maintain $n$ constant

# Curse of Dimensionality



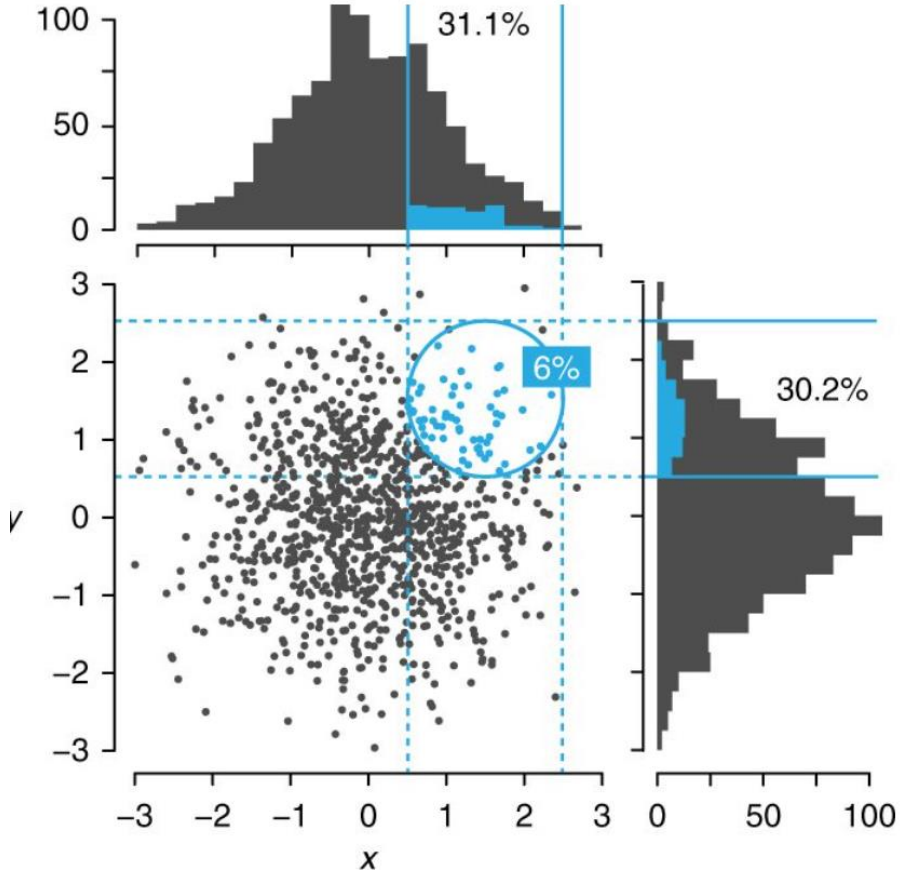

$$E(P(x_i \in \text{bin}_k)) \sim \frac{N}{k^M}$$

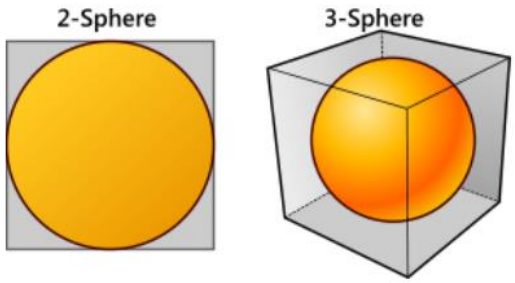
**Consequences:**

- with finite sample size (limited data collection), most of the cells are empty if the feature dimension is too high
- The estimation of probability density is unreliable

# Curse of Dimensionality

## Concentration of distances



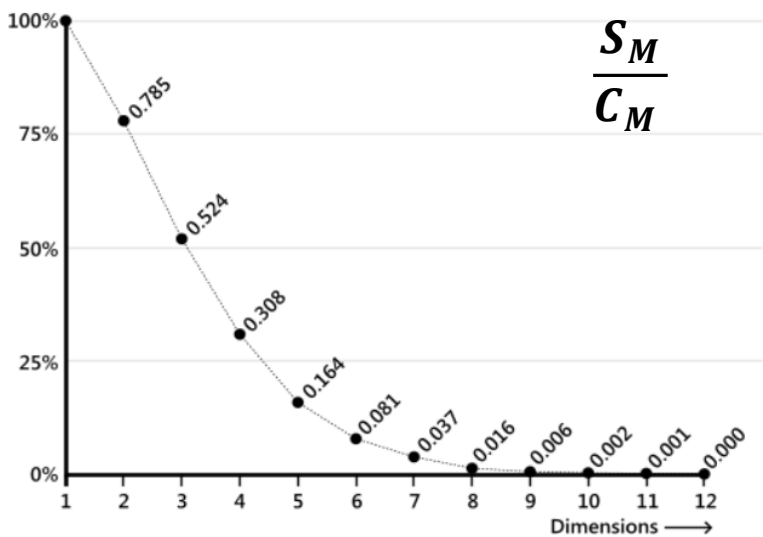



# Curse of Dimensionality

Volume of the hypersphere $$S_M(R) = \frac{\pi^{M/2}}{\Gamma(\frac{M}{2}+1)} R^M \sim \frac{1}{\sqrt{M\pi}}\left(\frac{2\pi e}{M}\right)^{\frac{M}{2}} R^M$$

(Using Stirling's approximation)

Volume of the hypercube $$C_M(R) = (2R)^M$$



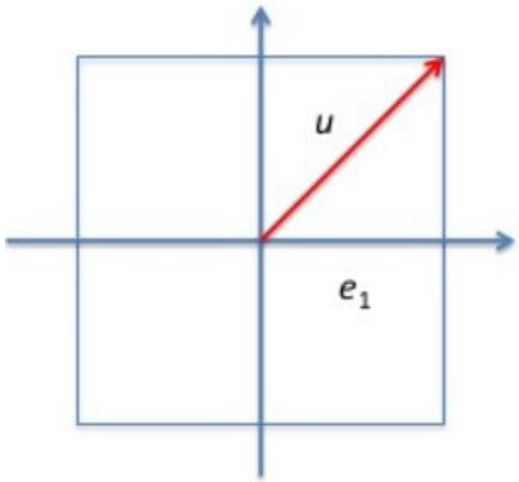

## Consequences :

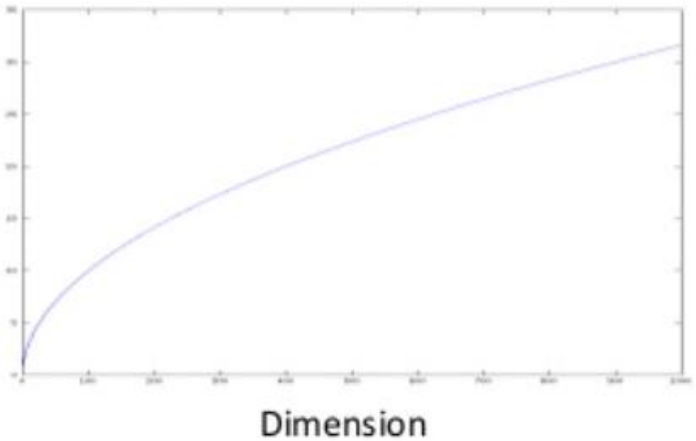
- The unit sphere becomes an insignificant volume relative to that of the unit cube.

In other words, almost all of the high-dimensional space is far away from the center.

# High dimensional diagonals

**In $M$ dimension, the unit hypercube has as diagonal $u = [1\ 1\ 1\ \ldots 1\ ]^T$, then**



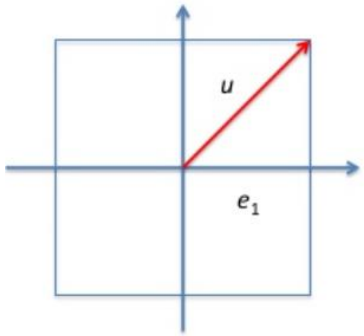




$$||\boldsymbol{u}||_2 = \sqrt{M} \quad \rightarrow \infty \quad \text{as} \quad M \rightarrow \infty$$

$$\boldsymbol{cos}\,(\boldsymbol{\theta_M}) = \boldsymbol{cos}(\boldsymbol{u}, \boldsymbol{e_1}) = \frac{\boldsymbol{u^T e_1}}{||\boldsymbol{u}||_2} = \frac{\boldsymbol{1}}{||\boldsymbol{u}||_2} \quad \rightarrow 0$$

# High dimensional diagonals

**In $M$ dimension, the unit hypercube has as diagonal $u = [1\ 1\ 1\ \ldots 1\ ]^T$, then**




$$\boldsymbol{\theta_M} \rightarrow \frac{\boldsymbol{\pi}}{\mathbf{2}}$$

- In high dimensions, all $(2^{M-1})$ diagonal vectors are orthogonal to the basis vectors
- High dimensional spaces have an exponential number of dimensions
- Everything along the diagonals is projected onto the origin

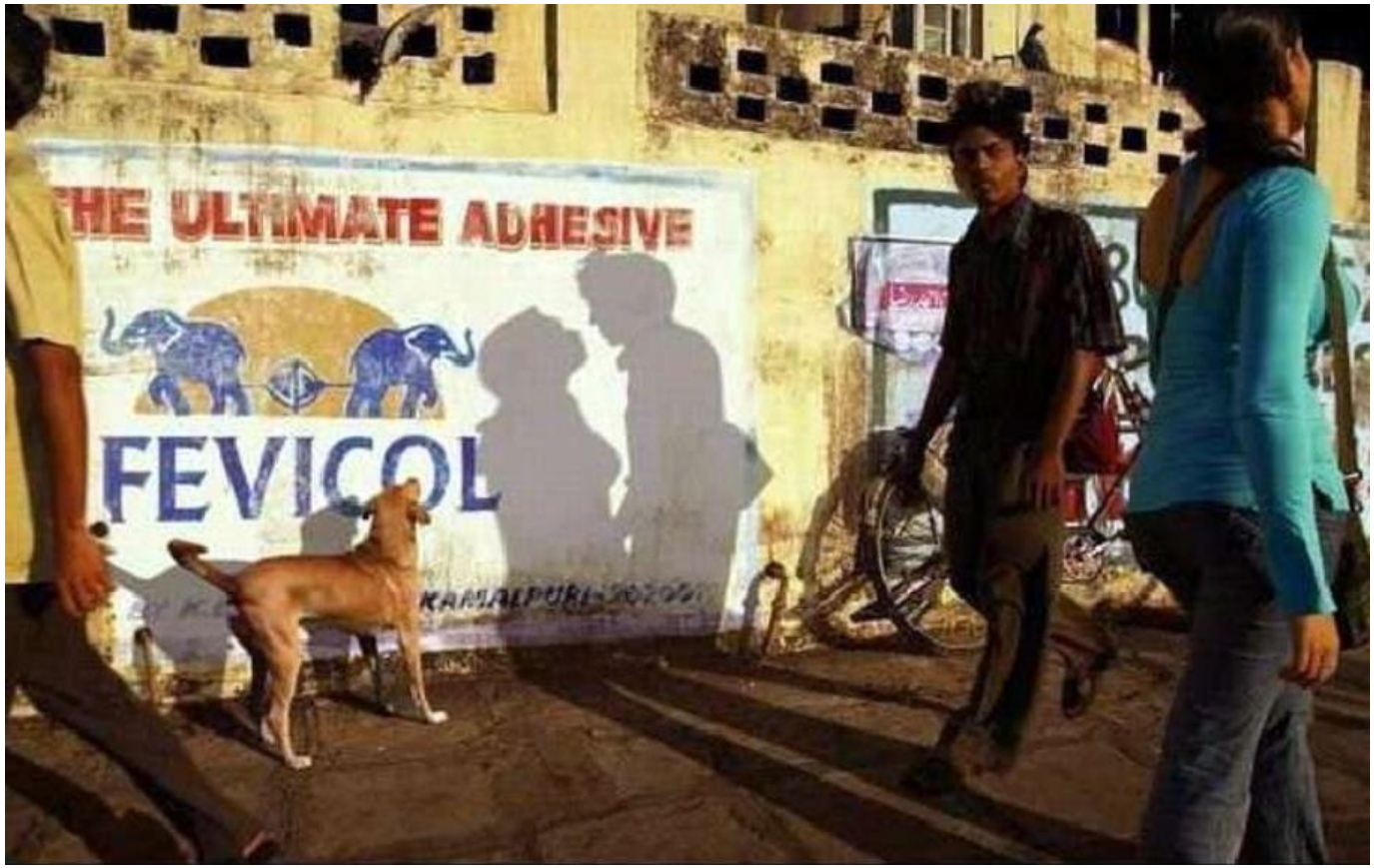
# Dimensionality Reduction

## Purpose

- Avoid curse of dimensionality
- Reduce amount of time and memory required by data mining algorithms
- Allow data to be more easily visualized
- May help to eliminate irrelevant features or reduce noise

**Feature Selection??**

# Dimensionality Reduction




While dimensionality reduction is an important tool in machine learning, we must always be aware that it can distort the data in misleading ways

# Dimensionality Reduction

## Feature Extraction

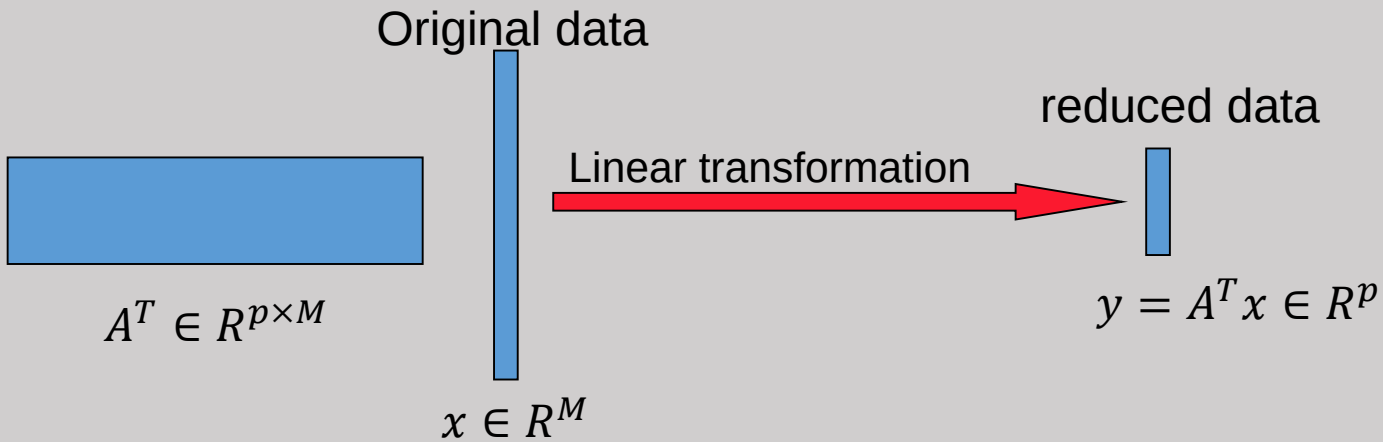
## Some Techniques

- **PCA (Principle Component Analysis)**
- **LDA (Linear Discriminant Analysis)**
- Singular Value Decomposition
- Others : supervised and non-linear techniques

# Dimensionality Reduction

Feature reduction refers to the **mapping of the original high dimensional data into a lower dimensional space**

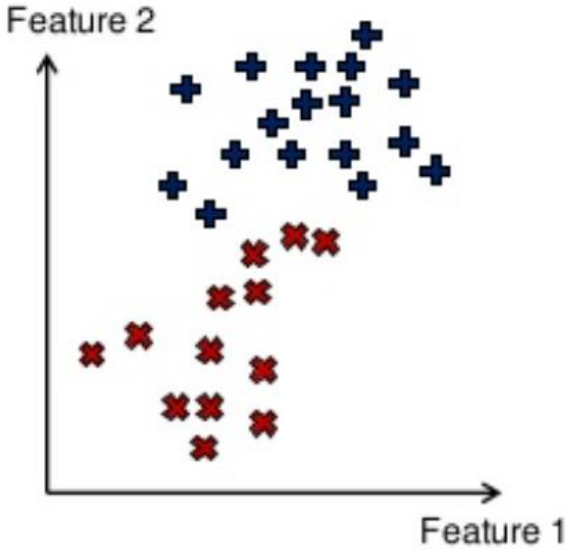
Given a set of data of $M$ variables $\{x_1, x_2, \dots, x_n\}$ compute the linear transformation (Projection)




Criterion for feature reduction can be different based on different problem settings

- Unsupervised setting: minimize the information loss
- Supervised setting: maximize the class discrimination

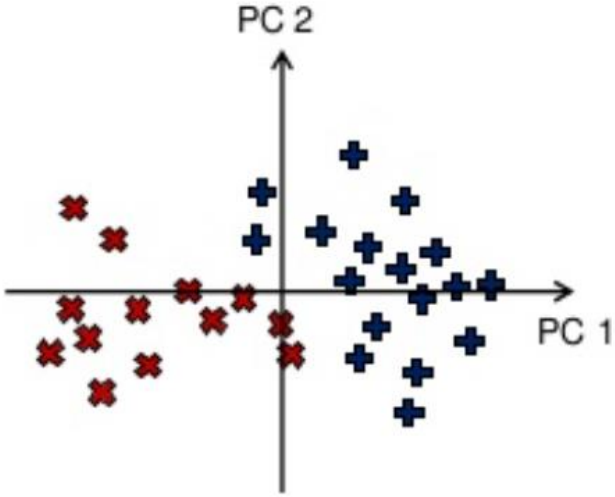
# Dimensionality Reduction




PCA

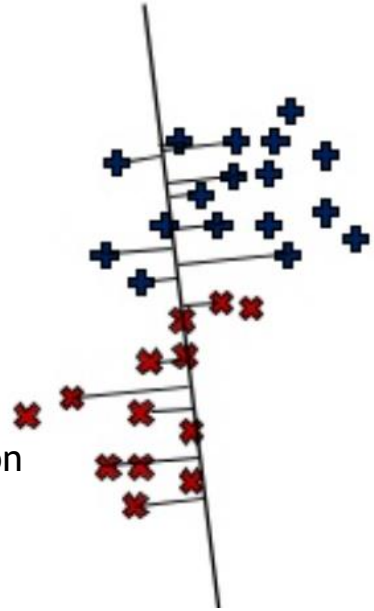
Unsupervised setting:

minimize the information loss




LDA

Supervised setting:
maximize the class discrimination



# Linear Algebra Background

# Eigenvalues & Eigenvectors

- **Eigenvector equation** (for a square $m \times m$ matrix $\mathbf{S}$)

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

(right) eigenvector $\mathbf{v} \in \mathbb{R}^m \neq \mathbf{0}$

eigenvalue $\lambda \in \mathbb{R}$

*Example*

$$\begin{pmatrix} 6 & -2 \\ 4 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- How many eigenvalues are there at most?

$$\mathbf{S}\mathbf{v} = \lambda \mathbf{v} \iff (\mathbf{S} - \lambda \mathbf{I})\, \mathbf{v} = \mathbf{0}$$

only has a non-zero solution if $|\mathbf{S} - \lambda \mathbf{I}| = 0$

this is a $m$-th order equation in $\lambda$ which can have **at most $m$ distinct solutions** (roots of the characteristic polynomial)

# Eigenvalues & Eigenvectors

- Eigenvectors for distinct eigenvalues are **orthogonal**

$$\mathbf{S}\mathbf{v}_{\{1,2\}} = \lambda_{\{1,2\}}\mathbf{v}_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow \langle \mathbf{v}_1, \mathbf{v}_2 \rangle = 0$$

- All eigenvalues of a real symmetric matrix are **real**.

$$\text{for } \lambda \in \mathbb{C}, \text{ if } |\mathbf{S} - \lambda\mathbf{I}| = 0 \text{ and } \mathbf{S} = \mathbf{S}' \Rightarrow \lambda \in \mathbb{R}$$

- All eigenvalues of a positive semidefinite matrix are **non-negative**

$$\mathbf{w}'\mathbf{S}\mathbf{w} \geq 0, \ \forall \mathbf{w} \in \mathbb{R}^d, \text{ then if } \mathbf{S}\mathbf{v} = \lambda\mathbf{v} \Rightarrow \lambda \geq 0$$

# Eigen Decomposition

- Let $\mathbf{S} \in \mathbb{R}^{m \times m}$ be a **square** matrix with ***m* linearly independent eigenvectors** (a non-defective matrix)

- **Theorem**: Exists a (unique) **eigen decomposition**
  (cf. matrix diagonalization theorem)

*diagonal*

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{-1}$$

*similarity transform*

- Columns of $\mathbf{U}$ are **eigenvectors** of $\mathbf{S}$

- Diagonal elements of $\boldsymbol{\Lambda}$ are **eigenvalues** of $\mathbf{S}$

$$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_m), \quad \lambda_i \geq \lambda_{i+1}$$

# Symmetric Eigen Decomposition

- If $\mathbf{S} \in \mathbb{R}^{m \times m}$ is a **symmetric** matrix:

- **Theorem**: Exists a (unique) **eigen decomposition**

$$\mathbf{S} = \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^{\mathbf{T}}$$

  - where $\mathbf{U} \in \mathbb{R}^{m \times m}$ is **orthogonal**

$$\begin{cases} \mathbf{U}' = \mathbf{U}^{-1} \\ \langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{ij} \end{cases}$$

*columns are orthogonal and length normalized*

# Spectral Decomposition

- **Spectral decomposition theorem** (finite dimensional, symmetric case, in general: normal matrices/operators)
- Eigenvalue subspaces

$$\mathcal{U}_\lambda = \{\mathbf{u} : \mathbf{S}\mathbf{u} = \lambda\mathbf{u}\} = \ker(\mathbf{S} - \lambda\mathbf{I})$$

- Direct sum representation

$$\mathbb{R}^m = \bigoplus_{\lambda \in \lambda(\mathbf{S})} \mathcal{U}_\lambda$$

# Singular Value Decomposition

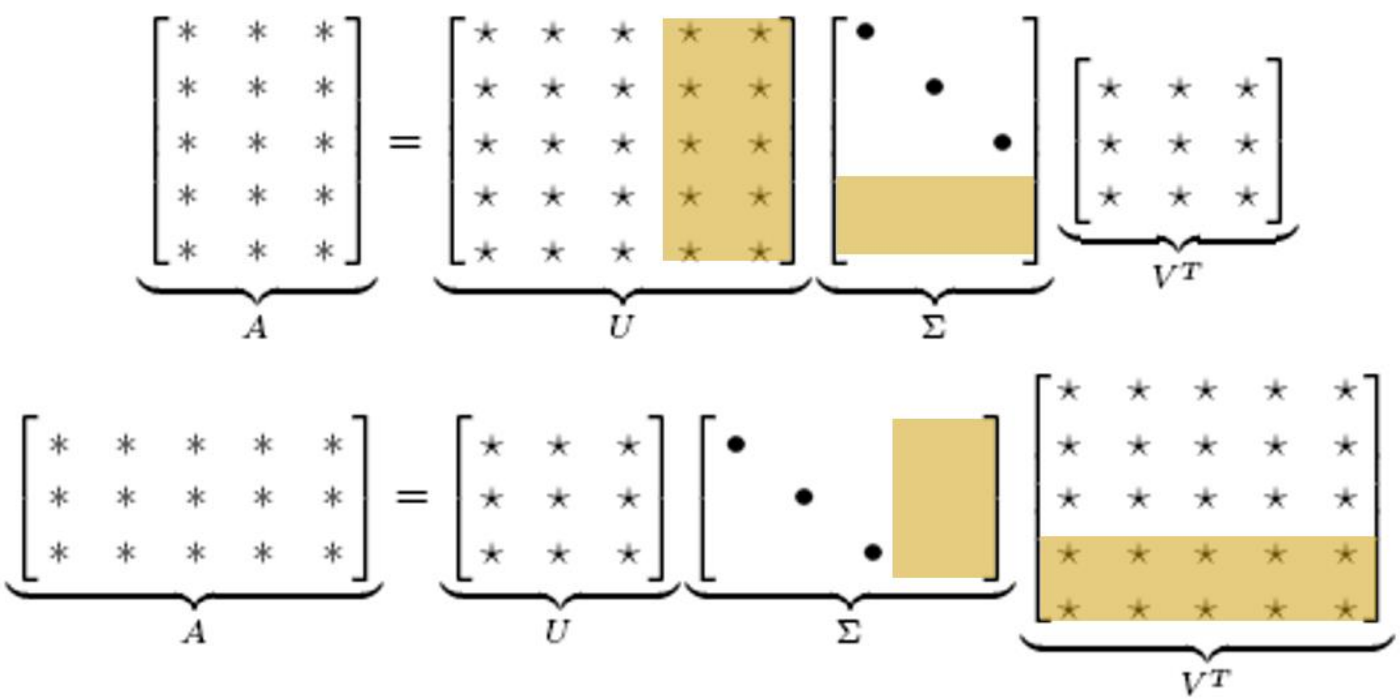
- For an arbitrary matrix $\mathbf{A}$ there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\,\mathbf{V}^{\mathbf{T}} \in \mathbb{R}^{n\times m}$$

- Where
  - (i) $\mathbf{U} \in \Re^{n\times k}$ $\quad \boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ $\quad \mathbf{V} \in \mathbb{R}^{m\times k}$
  - (ii) $\mathbf{U}'\mathbf{U} = \mathbf{I}$ $\quad \mathbf{V}'\mathbf{V} = \mathbf{I}$ *orthonormal columns*
  - (iii) $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_k),\ \sigma_i \geq \sigma_{i+1}$ *singular values (ordered)*
  - (iv) $k = \mathrm{rank}(\mathbf{A})$

# Singular Value Decomposition

- Illustration of SVD dimensions and sparseness
- **Full SVD** (padded with zeros) vs. **reduced SVD**



# Low-rank Approximation

- SVD can be used to compute optimal **low-rank approximations.**

- Approximation problem:

$$\mathbf{X}^* = \underset{\hat{\mathbf{X}}:\mathrm{rank}(\mathbf{X})=\mathrm{q}}{\mathrm{argmin}} \|\mathbf{X} - \hat{\mathbf{X}}\|_F$$

*Frobenius norm*

$$\|\mathsf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}\,.$$

- Solution via SVD

$$\mathbf{X}^* = \mathbf{U}\,\mathrm{diag}(\sigma_1, \ldots, \sigma_q, \underbrace{0, \ldots, 0}_{\text{set small singular values to zero}})\,\mathbf{V}'$$

$$\mathbf{X}^* = \sum_{r=1}^{q} \sigma_r \mathbf{u}_r \mathbf{v}_r'$$

*column notation: sum of rank 1 matrices*

# Pattern Matrix

- Statistics and machine learning typically starts from data given in the form of **observations**, **feature vectors** or **patterns**

- Feature vectors (in some m-dimensional Euclidean space)

$$\mathbf{x_i} \in \mathcal{X} \subseteq \mathbb{R}^m, \quad i = 1, \ldots, n$$

- Patterns can be summarizes into the **pattern matrix**

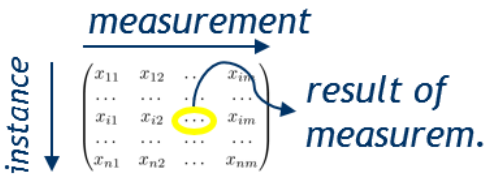
$$\mathbf{X} \in \mathbb{R}^{n \times m}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1' \\ \ldots \\ \mathbf{x}_i' \\ \ldots \\ \mathbf{x}_n' \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{im} \\ \ldots & \ldots & \ldots & \ldots \\ x_{i1} & x_{i2} & \ldots & x_{im} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix}$$

*transposed i-th pattern*

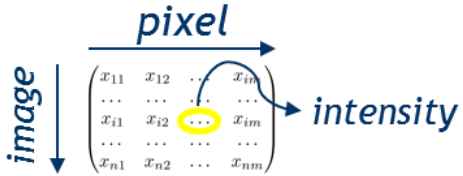
# Examples: Pattern Matrices

$$\mathbf{X} \in \mathbb{R}^{n \times m}$$

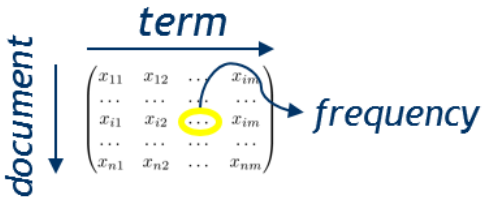
- **Measurement** vectors
  - $i$: instance number, e.g. a house
  - $j$: measurement, e.g. the area of a house



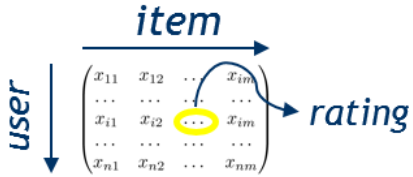

- **Digital images** as gray-scale **vectors**
  - $i$: image number
  - $j$: pixel value at location $j=(k,l)$




- **Text documents** in **bag-of-words** representation
  - $i$: document number
  - $j$: term (word or phrase) in a vocabulary




- **User rating** data
  - $i$: user number
  - $j$: item (book, movie)



# Sample Covariance Matrix

- **Mean pattern** and **centered patterns**

$$\bar{\mathbf{x}} \equiv \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \quad \tilde{\mathbf{x}}_i \equiv \mathbf{x}_i - \bar{\mathbf{x}}, \quad \tilde{\mathbf{X}} \equiv \begin{pmatrix} \tilde{\mathbf{x}}_1' \\ \tilde{\mathbf{x}}_2' \\ \dots \\ \tilde{\mathbf{x}}_n' \end{pmatrix} = \mathbf{X} - \mathbf{1}_n\bar{\mathbf{x}}'$$

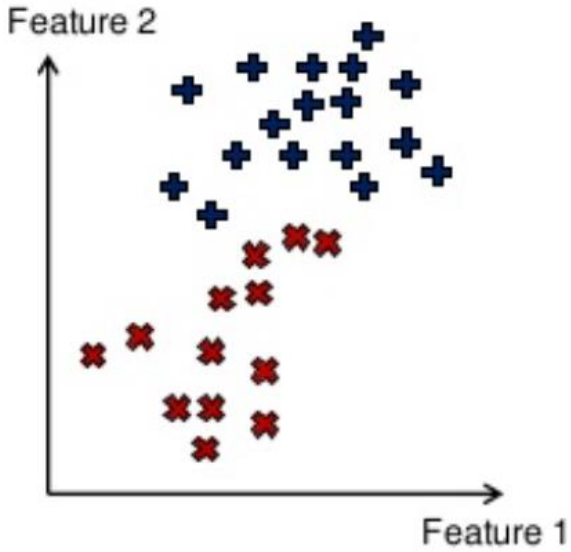
- **Sample covariance matrix** measures (empirical) correlations between different features or dimensions

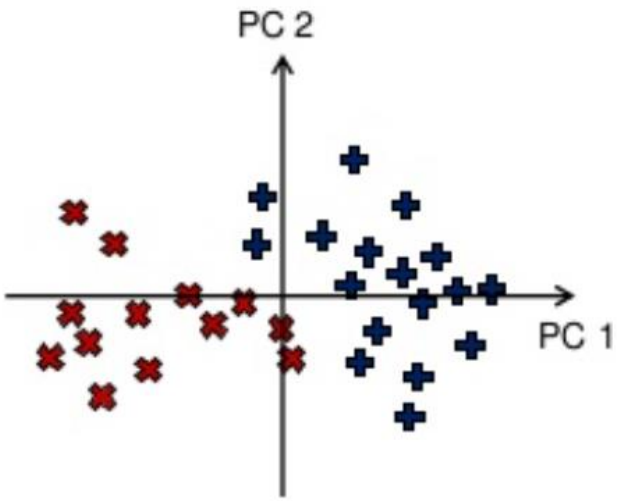
$$\mathbf{S} \in \mathbb{R}^{m\times m}, \quad \mathbf{S} = (S_{rs})_{1\le r,s\le m}, \quad S_{rs} = \frac{1}{n}\sum_{i=1}^{n}\tilde{x}_{ir}\tilde{x}_{is}$$

in terms of the pattern matrix $\boxed{\mathbf{S} = \frac{1}{n}\tilde{\mathbf{X}}^T\tilde{\mathbf{X}}}$

# PCA

## Principle Component Analysis



# Principle Components Analysis

**Idea:**

- Given data points in a $M$-dimensional space, project into lower dimensional space while preserving as much information as possible

  Eg,

  find best planar approximation to 3D data

  find best 12-D approximation to $10^4$-D data

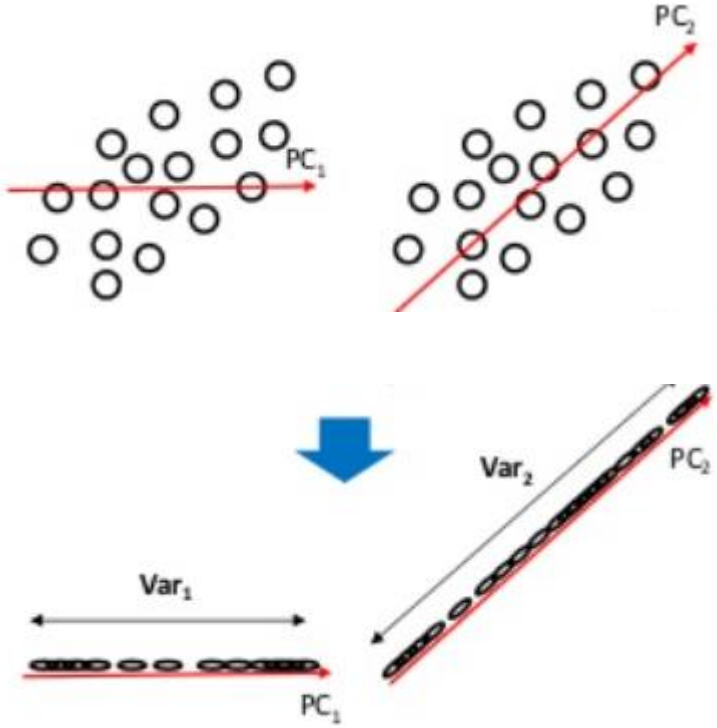
- In particular,

  (1) choose projection that *minimizes squared error* in reconstructing original data

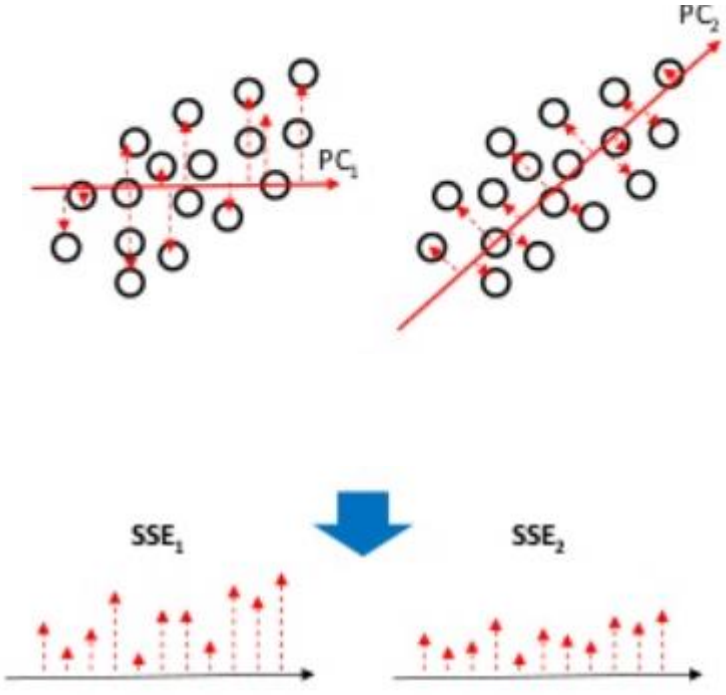
  (2) choose projection that *maximizes the variance* of the projected data

# Principle Components Analysis

## 1) Maximum variance



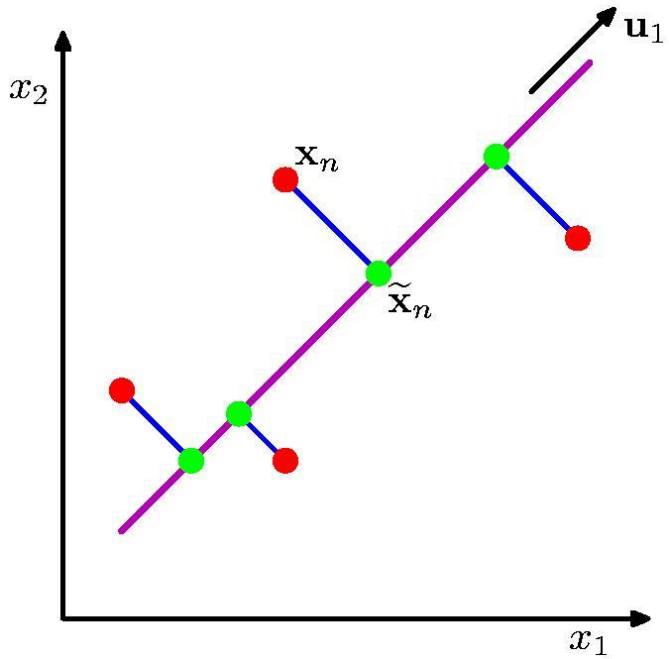

## 2) Minimum error




SSE: Sum of squared errors

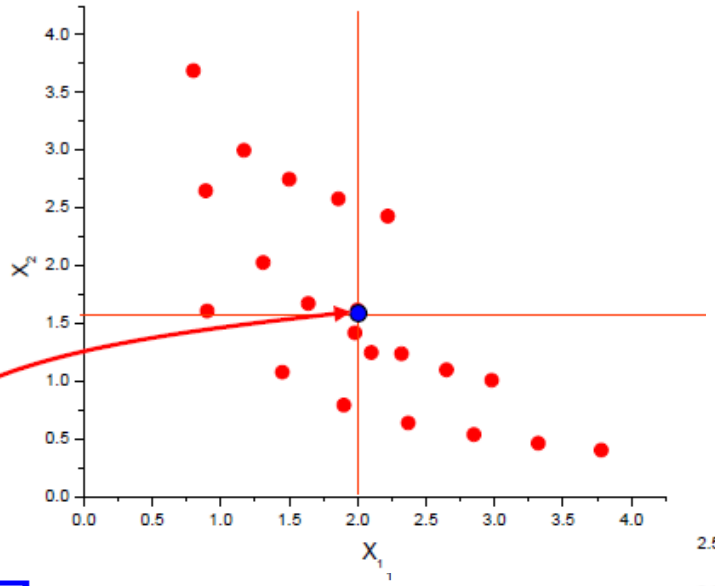
# Principle Components Analysis

Given data points in a $M$-dimensional space, for large $M$, how does one project on to a 1 dimensional space?

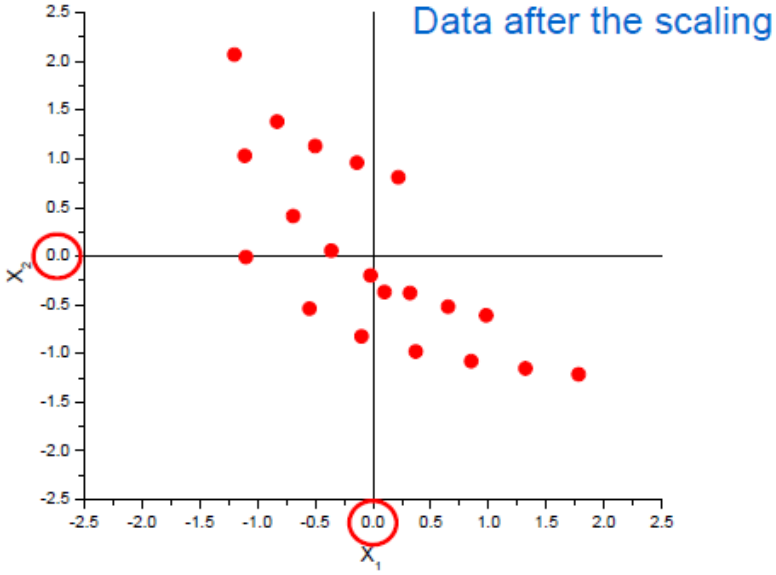



Choose a line that fits the data so the points are spread out well along the line

# Mean centering

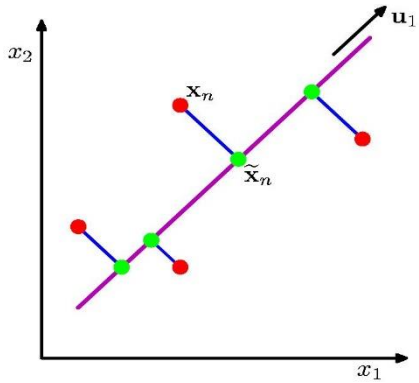


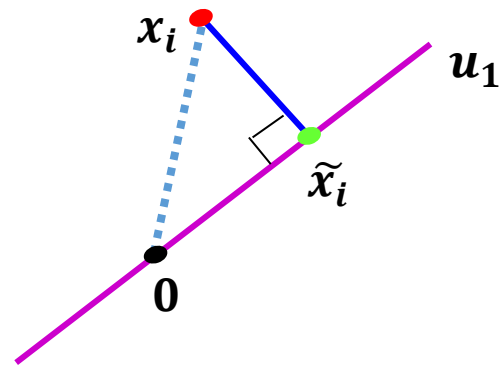

| Avg. of $X_1$ | Avg. of $X_2$ |
|---|---|
| ↓ | ↓ |
| (1.9995, | 1.618) |



# Principle Components Analysis

Given the **centered** data $\{x_1, x_2, \ldots, x_m\}$, compute the principal vector:

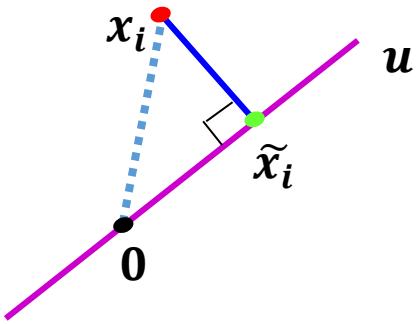



- maximizes variance of projected data (purple line)

- minimizes mean squared distance between data point and projections (sum of blue lines)

$$\sum ||x_i||^2 = \sum (||\widetilde{x}_i||^2 + ||x_i - \widetilde{x}_i||^2) = \sum ||\widetilde{x}_i||^2 + \sum ||x_i - \widetilde{x}_i||^2$$




# Principle Components Analysis

$$\tilde{x}_i = \frac{u^T x_i}{u^T u} u = (u^T x_i) u$$

$$||u|| = 1$$

1st PCA vector

$$u_1 = \underset{||u||=1}{arg\max} \sum_{i=1}^{m} ||(u^T x_i) u \;||^2 = \underset{||u||=1}{arg\max} \sum_{i=1}^{m} (u^T x_i)^2$$

$$\sum_{i=1}^{m} (u^T x_i)^2 = (u^T X)(u^T X)^T = u^T X X^T u = u^T (m\Sigma) \mathrm{u}$$

$$X = (x_1 \mid x_2 \mid \ldots \mid x_m)$$

$\Sigma = \frac{1}{m} X X^T = \frac{1}{m} \sum (x_i \; x_i^T)$ : covariance matrix of $\{x_1, x_2, \ldots, x_m\}$,

$$u_1 = \underset{||u||=1}{arg\max} \; u^T \Sigma \, u$$

# Lagrange Multipliers

Maximize $f(x, y)$ s.t. $g(x, y) = c$

Need both $f$ and $g$ to have continuous partial derivatives.

We introduce a new variable $\lambda$, called a *Lagrange multiplier*, and the *Lagrangian* $L$ defined by

$$L(x, y, \lambda) = f(x, y) - \lambda[g(x, y) - c]$$

Note that if $f(x_0, y_0)$ is a maximum of $f(x, y)$,
then $\exists \lambda_0$ s.t. $(x_0, y_0, \lambda_0)$ is a stationary point for the $L$,
where the partial derivatives of $L$ are zero.

# Lagrange Multipliers

$$u_1 = \arg\max_{\|u\|=1} u^T \Sigma u$$

- **Lagrange multiplier** technique

$$\mathcal{L}(\mathbf{u}, \lambda) = \langle \mathbf{u}' \boldsymbol{\Sigma} \mathbf{u} + \lambda(\langle \mathbf{u}, \mathbf{u} \rangle - 1)$$

$\downarrow$ *differentiation*

$$(\boldsymbol{\Sigma} - \lambda \mathbf{I})\, \mathbf{u} = 0 \iff$$ *eigenvalue/vector equation*

- The solution must be an **eigenvector**. Which one?

$$\mathbf{u}' \boldsymbol{\Sigma} \mathbf{u} = \lambda \langle \mathbf{u}, \mathbf{u} \rangle = \lambda$$

*eigenvector* *length one*

- The solution is the **principal** eigenvector (i.e. the one with the largest eigenvalue)

# Principle Components Analysis

1st PCA vector $\quad \boldsymbol{u_1} = \underset{\|\boldsymbol{u}\| = \mathbf{1}}{\boldsymbol{arg}\,\mathbf{max}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{m}} \left(\boldsymbol{u^T x_i}\right)^{\mathbf{2}}$

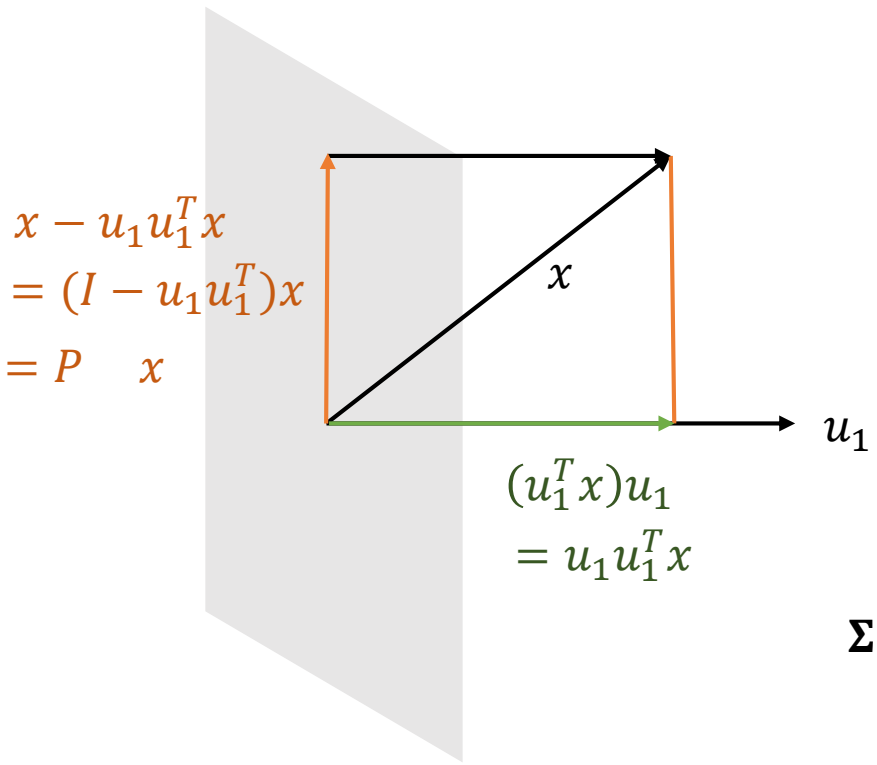
$$\parallel$$

$$\boldsymbol{u_1} = \underset{\|\boldsymbol{u}\| = \mathbf{1}}{\boldsymbol{arg}\,\mathbf{max}}\ \boldsymbol{u^T \Sigma\, u}$$

$$\parallel$$

1st eigenvector of $\boldsymbol{\Sigma}$

# Projection matrix




$$Pu_1 = (I - u_1 u_1^T) u_1 = u_1 - u_1 = 0$$

$$Pu_k = (I - u_1 u_1^T) u_k = u_k - 0 = u_k$$

$$\boldsymbol{\Sigma} = \frac{\mathbf{1}}{\boldsymbol{m}} \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$$

$$= \lambda_1 \, u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_M u_M u_M^T$$

$$\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^{\boldsymbol{T}} = \boldsymbol{P}(\lambda_1 u_1 u_1^T + \lambda_2 u_2 u_2^T + \cdots + \lambda_M u_M u_M^T)\boldsymbol{P}^{\boldsymbol{T}}$$

$$= \lambda_1 \boldsymbol{P} u_1 u_1^T \boldsymbol{P}^{\boldsymbol{T}} + \lambda_2 \boldsymbol{P} u_2 u_2^T \boldsymbol{P}^{\boldsymbol{T}} + \cdots + \lambda_M \boldsymbol{P} u_M u_M^T \boldsymbol{P}^{\boldsymbol{T}}$$

$$= \lambda_2 u_2 u_2^T + \cdots + \lambda_M u_M u_M^T$$

$u_2$ is the 1st eigenvector of $\boldsymbol{P}\boldsymbol{\Sigma}\boldsymbol{P}^{\boldsymbol{T}}$

# Principle Components Analysis

2nd PCA vector $\quad \boldsymbol{u_2} = \underset{\|\boldsymbol{u}\| = 1}{\boldsymbol{arg}\,\max} \sum_{i=1}^{m} ||\left(\boldsymbol{u}^T(x_i - u_1 u_1^T x_i)\right)\boldsymbol{u} \quad ||^2$

$$= \underset{\|\boldsymbol{u}\| = 1}{\boldsymbol{arg}\,\max} \sum_{i=1}^{m} ||\left(\boldsymbol{u}^T(P x_i)\right)\boldsymbol{u} \quad ||^2$$

$$= \underset{\|\boldsymbol{u}\| = 1}{\boldsymbol{arg}\,\max} \sum_{i=1}^{m} \left(\boldsymbol{u}^T \boldsymbol{P} \boldsymbol{x}_i\right)^2$$

$$= \underset{\|\boldsymbol{u}\| = 1}{\boldsymbol{arg}\,\max} \quad \boldsymbol{u}^T \boldsymbol{P} \boldsymbol{\Sigma}\, \boldsymbol{P}^T \boldsymbol{u}$$

$$= \text{1st eigenvector of} \quad \boldsymbol{P} \boldsymbol{\Sigma}\, \boldsymbol{P}^T$$

$$= \text{2nd eigenvector of} \quad \boldsymbol{\Sigma}$$

# Principle Components Analysis

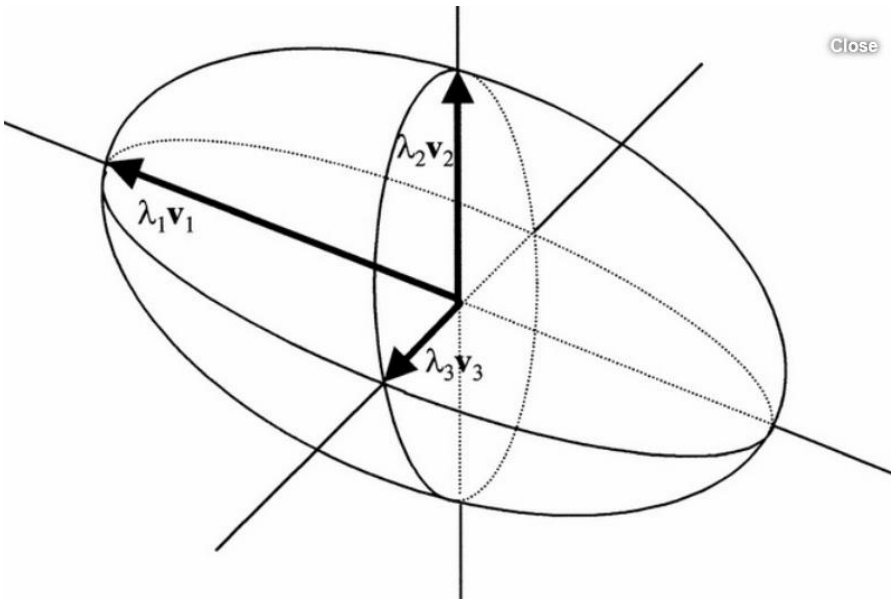
- To ensure that subsequent PCs are **uncorrelated**, search in the orthogonal complement of the directions identified so far. Spanned by remaining eigenvectors.

$$\mathrm{Span}(\mathbf{u}_1, \ldots, \mathbf{u}_{k-1}) \perp \mathrm{Span}(\mathbf{u}_k, \ldots, \mathbf{u}_d)$$

k$^{th}$ PCA vector $$\boldsymbol{u_k} = \underset{\|\boldsymbol{u}\| = 1}{\arg\max} \sum_{i=1}^{m} \left[ \boldsymbol{u}^T \left( \boldsymbol{x_i} - \sum_{j=1}^{k-1} \boldsymbol{u_j} \boldsymbol{u_j}^T \boldsymbol{x_i} \right) \right]^2$$

$=$ k$^{th}$ **eigenvector of** $\boldsymbol{\Sigma}$

# Principle Components Analysis




**PCA** basis vectors = the eigenvectors of $\boldsymbol{\Sigma}$

**Larger eigenvalue $\Rightarrow$ more important eigenvectors**

# Steps of PCA

(Step 1) **Mean centering**

Compute the average $\bar{x}$ of given data $\{x_1, x_2, \ldots, x_m\}$, subtracted from the data

(Step 2) **Calculation of Covariance matrix($\mathbf{S} = \frac{1}{\mathbf{m}}\mathbf{X}\mathbf{X}^{\mathbf{T}}$) of data**

**For symmetric matrices, their eigenvectors always are at right angles to each other**

**Eigenvectors of covariance matrix are orthogonal**

(Step 3) **Calculation of eigenvalues and eigenvectors of covariance matrix (S)**

**Eigenvectors**: the **orientations** of the principal axes of the ellipse

**Eigenvalues**: the **lengths** of each of the principal axes of ellipse

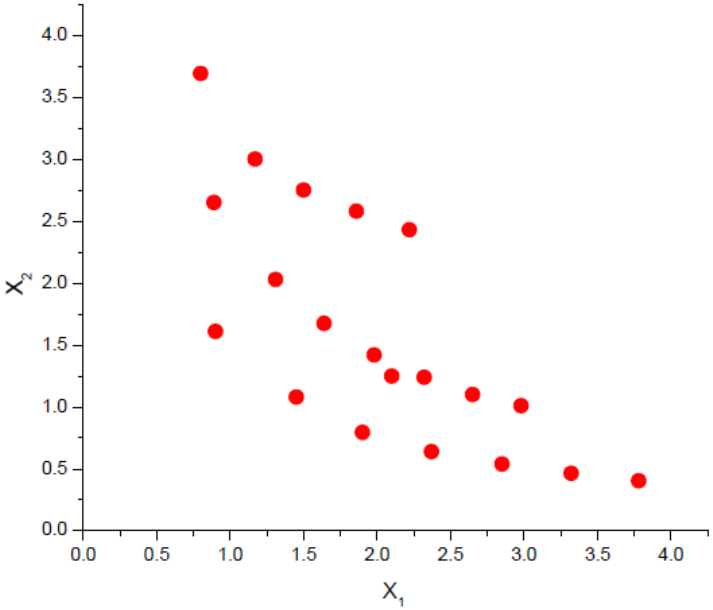
(Step 4) **Mathematical representations of transformation of axes**

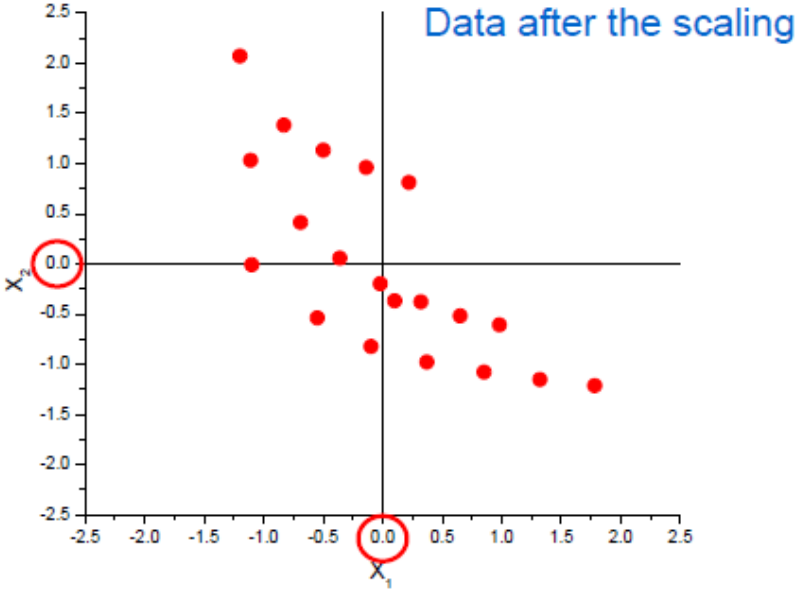
# Example of PCA

| Sample NO. | Element | Martynov-Batsanov's Electronegativity ($X_1$) | Zunger's pseudopotenial core radii sum ($X_2$) | Sample NO. | Element | Martynov-Batsanov's Electronegativity ($X_1$) | Zunger's pseudopotential core radii sum ($X_2$) |
|---|---|---|---|---|---|---|---|
| 1 | H | 2.1 | 1.25 | 11 | Al | 1.64 | 1.675 |
| 2 | Li | 0.9 | 1.61 | 12 | Si | 1.98 | 1.42 |
| 3 | Be | 1.45 | 1.08 | 13 | P | 2.32 | 1.24 |
| 4 | B | 1.9 | 0.795 | 14 | S | 2.65 | 1.1 |
| 5 | C | 2.37 | 0.64 | 15 | Cl | 2.98 | 1.01 |
| 6 | N | 2.85 | 0.54 | 16 | K | 0.8 | 3.69 |
| 7 | O | 3.32 | 0.465 | 17 | Ca | 1.17 | 3 |
| 8 | F | 3.78 | 0.405 | 18 | Sc | 1.5 | 2.75 |
| 9 | Na | 0.89 | 2.65 | 19 | Ti | 1.86 | 2.58 |
| 10 | Mg | 1.31 | 2.03 | 20 | V | 2.22 | 2.43 |

# Example of PCA

(Step 1) **Mean centering**




| Avg. of $X_1$ | Avg. of $X_2$ |
|---|---|
| ↓ | ↓ |
| (1.9995, | 1.618) |

# Example of PCA

(Step 2) **Calculation of Covariance matrix($S = \frac{1}{m}XX^T$) of data**

(Step 3) **Calculation of eigenvalues and eigenvectors of covariance matrix ($S$)**

$$S = \begin{bmatrix} 0.6881 & -0.5929 \\ -0.5929 & 0.9026 \end{bmatrix} \rightarrow \begin{vmatrix} 0.6881-\lambda & -0.5929 \\ -0.5929 & 0.9026-\lambda \end{vmatrix} = 0$$

Symmetrical

its eigenvectors are orthogonal (90°)

See next slide…

$$(0.6881-\lambda)(0.9026-\lambda)-(0.5929)^2 = 0$$

$$\lambda^2 - 1.5907\lambda + 0.27 = 0$$

$$\lambda = \frac{1.5907 \pm \sqrt{(1.5907)^2 - 4\times 0.27}}{2}$$

$$\boxed{\lambda_1 = 1.3978,\ \ \lambda_2 = 0.1928}$$

# Example of PCA

(Step 3) **Calculation of eigenvalues and eigenvectors of covariance matrix (*S*)**

For $\lambda_1 = 1.3978$, Eigenvectors: -0.6411 & 0.7675 or (0.6411 & -0.7675)

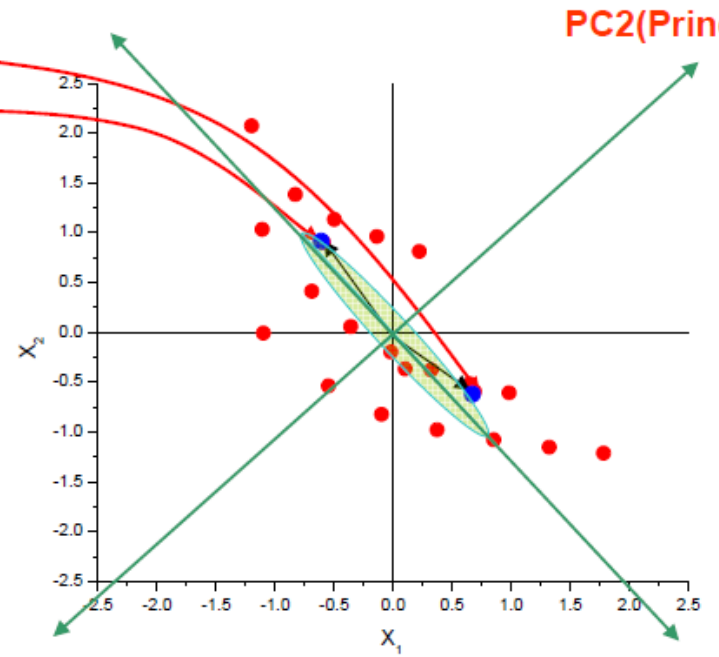
For $\lambda_2 = 0.1928$, Eigenvectors: -0.7675 & -0.6411 or (0.7675 & 0.6411)

Covariance matrix of scaled data matrix, ***S***

$$S = \begin{bmatrix} 0.6881 & -0.5929 \\ -0.5929 & 0.9026 \end{bmatrix}$$

Eigenvalues:

the lengths of each of the principal axes of ellipse



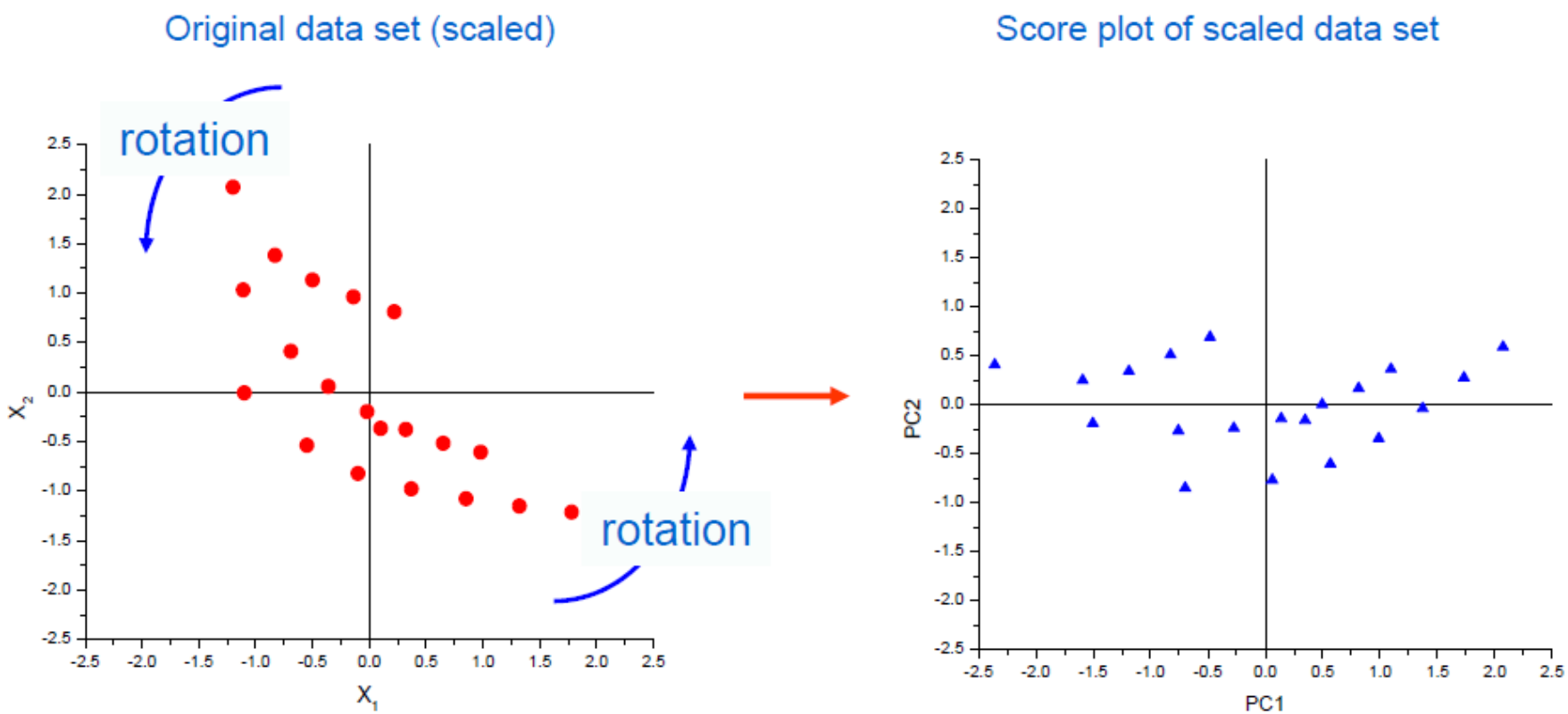

Orthogonal (90°)

Slope of minor axis =
ratio of eigenvector of the second largest eigenvalue($\lambda_2$)

Slope of major axis =
ratio of eigenvector of the largest eigenvalue($\lambda_1$)

# Example of PCA

(Step 4) **Mathematical representations of transformation of axes**



# LDA

## Linear Discriminant Analysis

# Linear discriminant analysis, two-classes

## Objective

LDA seeks to **reduce dimensionality while preserving as much of the class discriminatory information** as possible

Assume we have a set $M$-dimensional samples $\{x_1, x_2, \ldots, x_N\}$, $N_1$ of which belong to class $C_1$, and $N_2$ to class $C_2$.

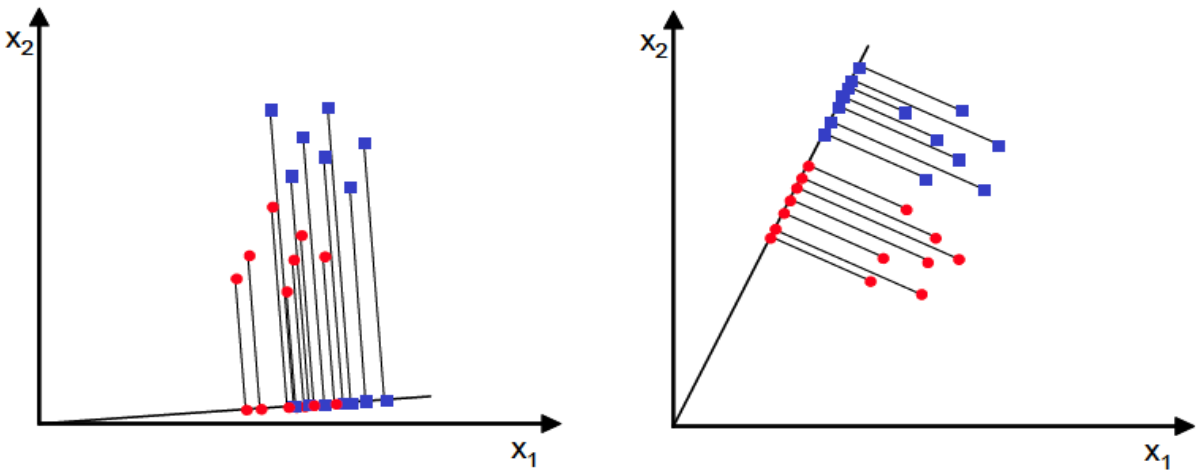
We seek to obtain a scalar $y$ by projecting the samples $x$ onto a line

$$y = u^T x$$

Of all the possible lines we would like to select the one that maximizes the separability of the scalars

# Linear discriminant analysis, two-classes

What is a good projection vector $u$?




**We need to define a measure of separation**

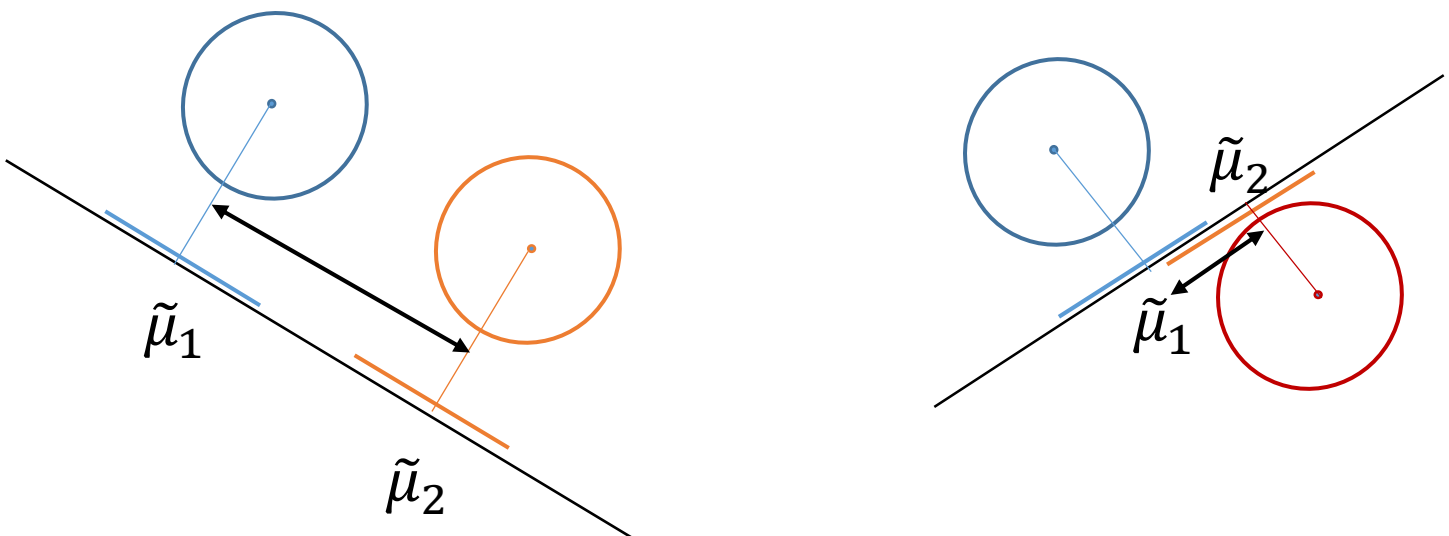
# Linear discriminant analysis, two-classes

The mean vector of each class in $x$-space and $y$-space is

$$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \qquad \tilde{\mu}_i = \frac{1}{N_i} \sum_{x \in C_i} y = \frac{1}{N_i} \sum_{x \in C_i} u^T x = u^T \mu_i$$

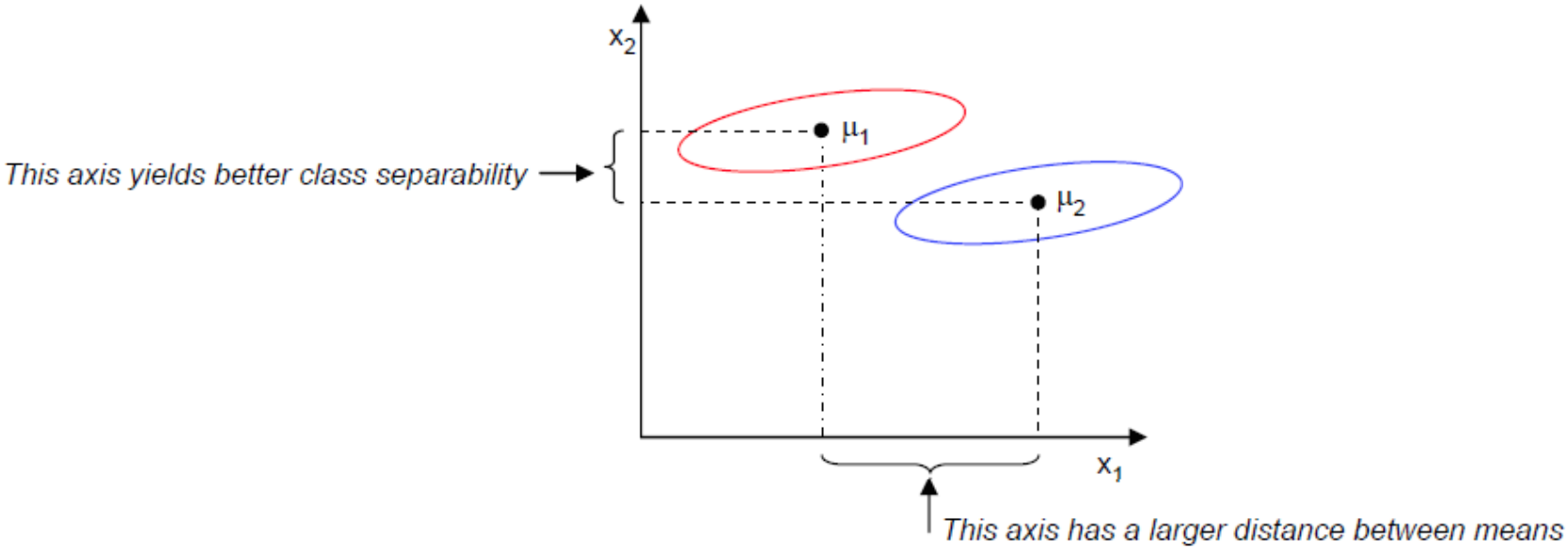
We could then choose the distance between the projected means as our objective function

$$J(w) = |\tilde{\mu}_1 - \tilde{\mu}_2| = |u^T(\mu_1 - \mu_2)|$$



# Linear discriminant analysis, two-classes

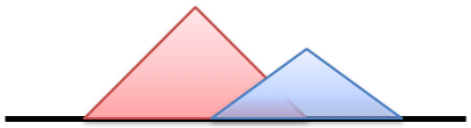
However, the distance between projected means is not good measure since it does not account for the standard deviation within classes



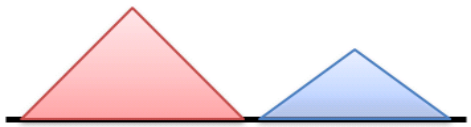
# Linear discriminant analysis, two-classes

A good projection satisfies the following conditions:

**The projected means are as farther apart as possible**

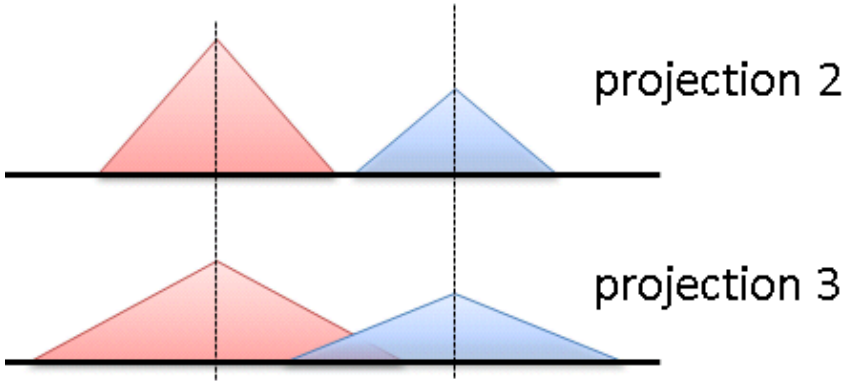


projection 1



projection 2

**Examples from the same class are projected very close to each other**



# Fisher's solution

Fisher suggested maximizing the difference between the means normalized by a measure of the within-class scatter

For each class we define the <u>scatter,</u> an equivalent of the variance, as

$$\tilde{s}_i{}^2 = \sum_{x \in C_i} (y - \tilde{\mu}_i)^2$$

$(\tilde{s}_1^2 + \tilde{s}_2^2)$ is called the **within-class scatter** of the projected examples

The Fisher linear discriminant is defined as the linear function $u^T x$ that maximizes the criterion function

$$J(u) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

$$J(u) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

# Within-class scatter

To find the optimum projection $u^*$,

we need **to express $J(u)$ as an explicit function of $u$**

We define a measure of the scatter in multivariate feature space $x$, which are scatter

$$S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \qquad S_W = S_1 + S_2$$

where $S_W$ is called the **within-class scatter matrix**

The scatter of the projection $y$ can then be expressed as a function of the scatter matrix in feature space $x$

$$\tilde{s}_i^2 = \sum_{x \in C_i} (y - \tilde{\mu}_i)^2 = \sum_{x \in C_i} (u^T x - u^T \mu_i)^2 = \sum_{x \in C_i} u^T (x - \mu_i)(x - \mu_i)^T u = u^T S_i u$$

$$\tilde{s}_1^2 + \tilde{s}_2^2 = u^T S_W u$$

$$J(u) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

# Between-class scatter

The difference between the projected means can be expressed in terms of the means in the original feature space

$$\begin{aligned}(\tilde{\mu}_1 - \tilde{\mu}_2)^2 &= u^T(\mu_1 - \mu_2)u^T(\mu_1 - \mu_2) \\ &= u^T\ (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T u \\ &= u^T\ S_B u\end{aligned}$$

The matrix $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$ is called the **between-class scatter.**

Note that, since $S_B$ is the outer product of two vectors, its rank is at most one.

The Fisher criterion function in term of $S_W$ and $S_B$

$$J(u) = \frac{u^T S_B u}{u^T S_W u}$$

# Fisher's linear discriminate

We need to solve

$$\max_{u \neq \mathbf{0}} \frac{u^T S_B u}{u^T S_W u}$$

$$\Updownarrow$$

$$\max u^T S_B u$$
$$\text{subject to } u^T S_W u = 1$$

Maximization problem
subject to an equality constraint

# Method of Lagrange Multipliers

# Lagrange Multipliers

$$L = u^T S_B u \; - \lambda[u^T S_W u - 1]$$

If $A$ is a symmetric matrix $\quad \nabla_u \, [u^T A u] = 2\, Au$.

Let $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$

Then $f(u) = u^T A u = a u_1^2 + (b + c) u_1 u_2 + d u_2^2$

$$\nabla f(u) = \begin{bmatrix} \dfrac{\partial f(u)}{\partial u_1} \\ \dfrac{\partial f(u)}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2a u_1 + (b + c) u_2 \\ (b + c) u_1 + 2d u_2 \end{bmatrix} = \begin{bmatrix} 2a & b + c \\ b + d & 2d \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = (A + A^T) u$$

$$\therefore \; \nabla_u \; [u^T A u] = 2\, Au$$

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$S_W = S_1 + S_2, \quad S_i = \textstyle\sum_{x \in C_i} \quad (x - \mu_i)(x - \mu_i)^T$$

are symmetric matrices.

$$\nabla_u \, L = 2(S_B - \lambda S_W) u$$

# Lagrange Multipliers

$$L = u^T S_B u \; - \lambda[u^T S_W u - 1]$$

$$\nabla_u L = 2(S_B - \lambda S_W)u \; = 0$$

$$S_B u = \lambda S_W u$$

The scalar λ in $S_B u = \lambda S_W u$ is ***a generalized eigenvalue* of the pair $(S_B, S_W)$** associated with the (generalized) eigenvector $u$.

# Generalized eigenvalue

The scalar λ is called ***a generalized eigenvalue*** **of the pair** $(\boldsymbol{A}, \boldsymbol{B})$ associated with the (generalized) eigenvector $v$

$$Av = \lambda Bv \qquad\qquad (A - \lambda B)v = \mathbf{0}$$

The expression $A - \lambda B$ with indeterminate $\lambda$ is called a ***matrix pencil***

The terms "matrix pair" and "matrix pencil" are used more or less interchangeably.

For example, if $v$ is any eigenvector for the pair $(A, B)$, then we also say that $v$ is any eigenvector of the matrix pencil $A - \lambda B$.

If B is nonsingular, then the generalized eigenvalue problem

is equivalent to a standard eigenvalue problem

$$Av = \lambda Bv \qquad\qquad B^{-1}Av = \lambda v$$

# Generalized eigenvalue

$S_B u = \lambda S_W u$

**Which generalized eigenvalue $\lambda$ should we choose?**

Recall that we want to solve $\max u^T S_B u$

subject to $u^T S_W u = 1$

Since the optimal $u$ satisfies $S_B u = \lambda S_W u$, we have

$$u^T S_B u = \lambda (u^T S_W u) \quad = \lambda$$

Hence we need to pick **the largest generalized eigenvalue and its corresponding generalized eigenvector.**

# Generalized eigenvalue

- $S_B v$ for any vector $v$, points in the same direction as $\mu_1 - \mu_2$

$$\because \quad S_B v = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T v \quad = (\mu_1 - \mu_2)\,\alpha$$

If $S_W$ is nonsingular, can convert this to a standard eigenvalue problem.

$$S_B u = \lambda S_W u \quad \Rightarrow \quad S_W^{-1} S_B u = \lambda u$$

Thus can solve the eigenvalue problem immediately

$$u = S_W^{-1}(\mu_1 - \mu_2)$$

$$\because \quad S_W^{-1}\, S_B u = \; S_W^{-1} S_B [S_W^{-1}(\mu_1 - \mu_2)] = S_W^{-1}[\alpha(\mu_1 - \mu_2)] = \alpha \quad S_W^{-1}(\mu_1 - \mu_2) = \alpha\, u$$

$$S_W = S_1 + S_2 \qquad S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

# LDA in 5 steps

(Step 1) Computing the $M$-dimensional mean vectors

(Step 2) Computing the Scatter Matrices

Within-class scatter matrix

$$S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

$$S_W = S_1 + S_2$$

Between-class scatter matrix

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

(Step 3) Solving the generalized eigenvalue problem for the matrix

(Step 4) Selecting linear discriminants for the new feature subspace

(Step 5) Transforming the samples onto the new subspace

# Example

Compute the LDA projection for the following 2D dataset

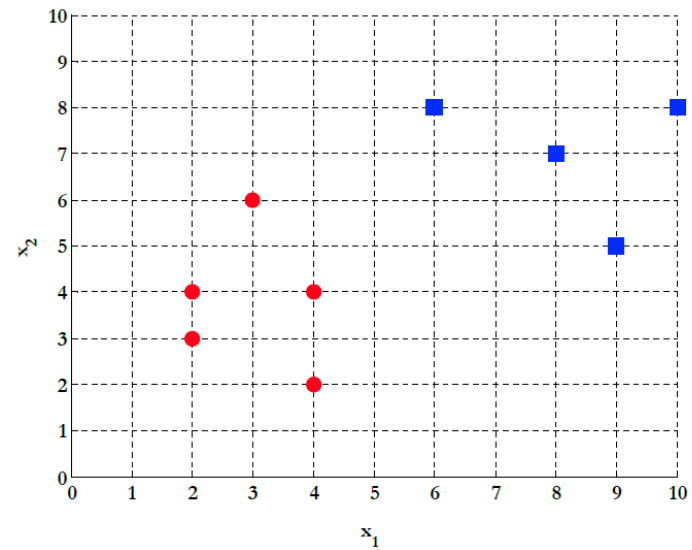
$$C_1 = \{(4,2),(2,4),(2,3),(3,6),(4,4)\}$$
$$C_2 = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$$

(Solution)

(Step 1) $\mu_1 = (3, 3.8),\ \mu_2 = (8.4, 7.6)$

(Step 2) $S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$

$$S_1 = \begin{bmatrix} 1 & -.25 \\ -.25 & 2.2 \end{bmatrix},\ S_2 = \begin{bmatrix} 2.3 & -.05 \\ -.05 & 3.3 \end{bmatrix}$$




The within class scatter $S_W = \begin{bmatrix} 3.3 & -.30 \\ -.30 & 5.5 \end{bmatrix}$,

between-class scatter $S_B = \begin{bmatrix} 29.16 & 20.52 \\ 20.52 & 14.44 \end{bmatrix}$

# Example

(Step 3) Solving the generalized eigenvalue problem $S_W^{-1} S_B u = \lambda u$

$$\left| S_W^{-1} S_B - \lambda I \right| = \begin{vmatrix} 9.2213 - \lambda & 6.489 \\ 4.2339 & 2.9794 - \lambda \end{vmatrix} = 0$$

$$\Rightarrow \lambda_1 = 12.2007 \qquad \lambda_2 = 0$$

$$\begin{bmatrix} 9.2213 & 6.489 \\ 4.2339 & 2.9794 \end{bmatrix} u_1 = 12.2007\, u_1$$

$$\begin{bmatrix} 9.2213 & 6.489 \\ 4.2339 & 2.9794 \end{bmatrix} u_2 = 0\, u_2$$

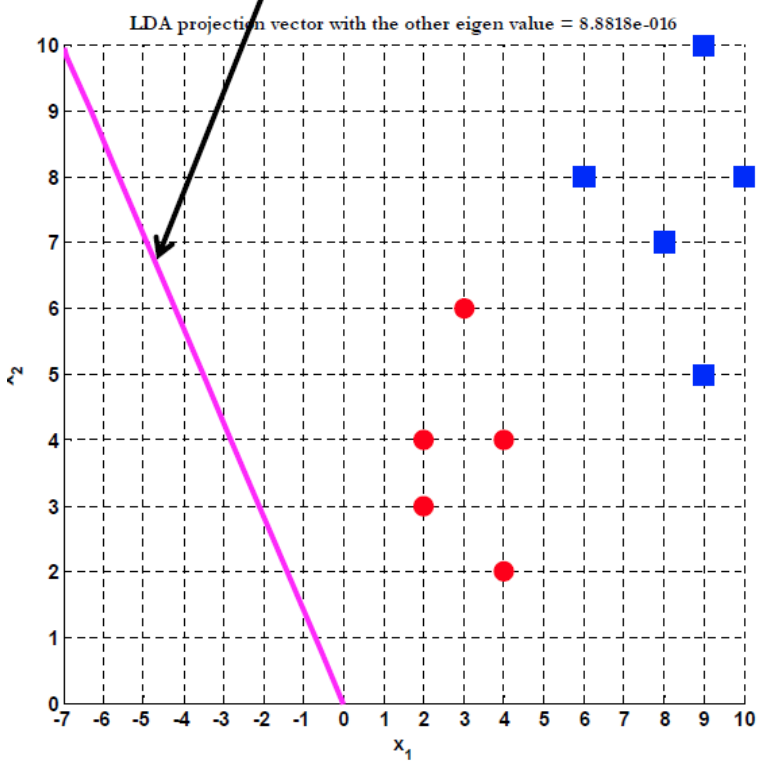
$$\Rightarrow u_1 = \begin{bmatrix} 0.9088 \\ 0.4173 \end{bmatrix}$$

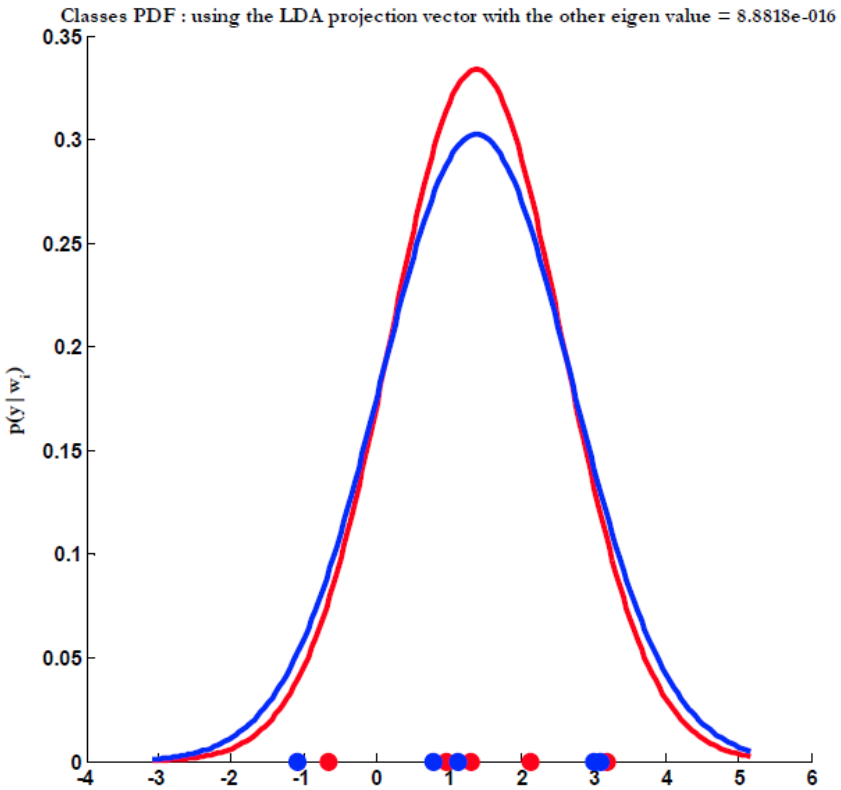
$$\Rightarrow u_2 = \begin{bmatrix} -0.5755 \\ 0.8178 \end{bmatrix}$$

# Example

$$\lambda_2 = 0 \qquad u_2 = \begin{bmatrix} -0.5755 \\ 0.8178 \end{bmatrix}$$

The projection vector corresponding to the **smallest** eigen value



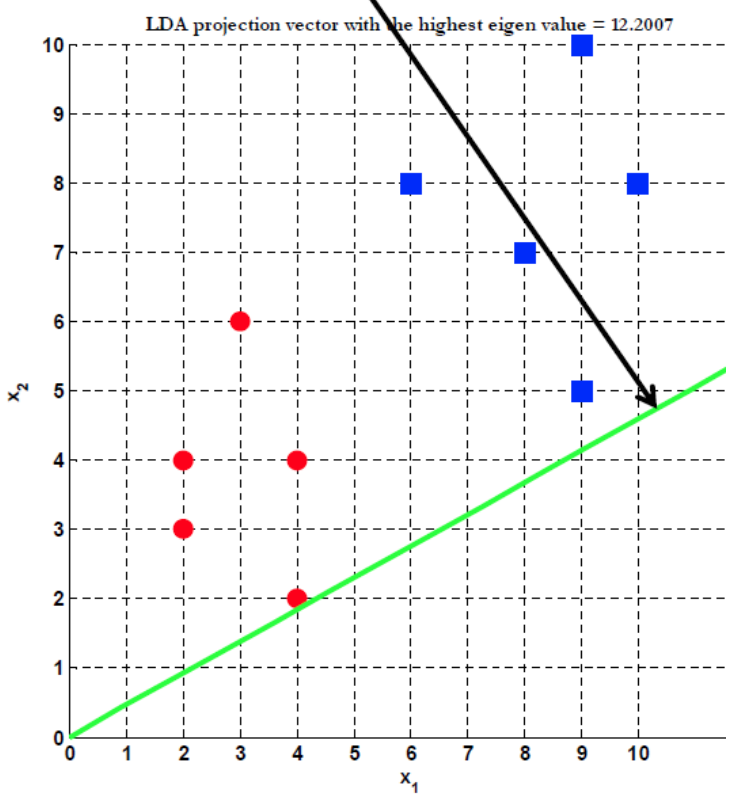



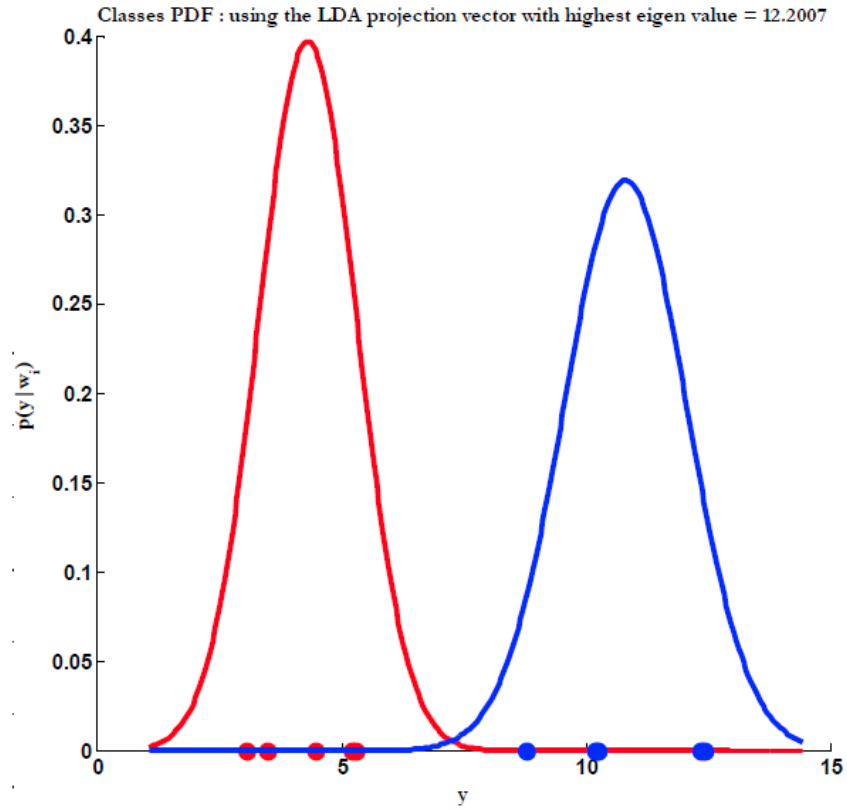

Using this vector leads to **bad separability** between the two classes

# Example

$\lambda_1 = 12.2007 \quad u_1 = \begin{bmatrix} 0.9088 \\ 0.4173 \end{bmatrix}$

The projection vector corresponding to the **highest** eigen value




Using this vector leads to **good separability** between the two classes

# LDA, C classes

**Fisher's LDA generalizes for C-class problems**

Instead of one projection $y$,

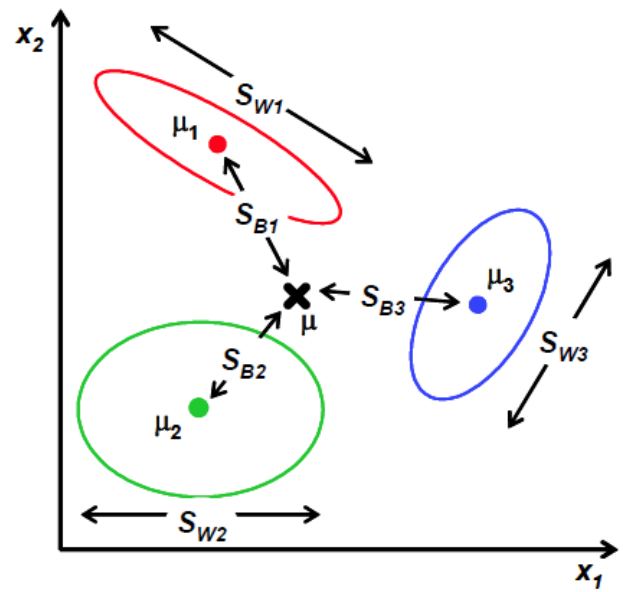
we will now seek $(C-1)$ projections $[y_1, y_2, \dots, y_{C-1}]$

by means of $(C-1)$ projection vectors $u_i$

arranged by columns into a projection matrix

$$W = [u_1 | u_2 | \dots | u_{C-1}]:$$

$$\boldsymbol{y_i = u_i^T x \;\Rightarrow\; y = W^T x}$$




**Derivation**

The within class scatter generalizes as $S_W = \sum_{i=1}^{C} S_i$

where $S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$ and $\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x$

The between-class scatter becomes

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Where $\mu = \frac{1}{N} \sum_{\forall x} x = \frac{1}{N} \sum_{i=1}^{C} \mu_i$

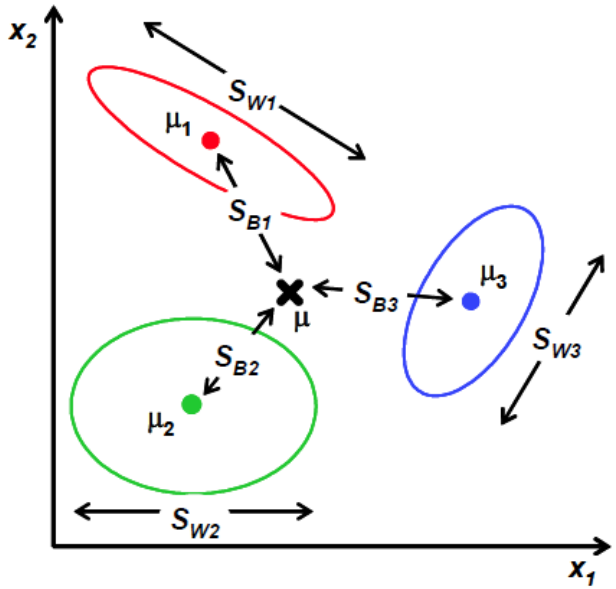
# LDA, C classes

- Similarly, we define the mean vector and scatter matrices for the projected samples as

$$\tilde{\mu}_i = \frac{1}{N_i} \sum_{x \in C_i} y \qquad \tilde{\mu} = \frac{1}{N} \sum_{\forall x} y$$

$$\tilde{S}_W = \sum_{i=1}^{C} \sum_{x \in C_i} (y - \tilde{\mu}_i)\,(y - \tilde{\mu}_i)^T$$

$$\tilde{S}_B = \sum_{i=1}^{C} N_i\,(\tilde{\mu} - \tilde{\mu}_i)(\tilde{\mu} - \tilde{\mu}_i)^T$$



# LDA, C classes

- From our derivation for the two-class problem, we can write

$$\tilde{S}_W = W^T S_W W \qquad\qquad \tilde{S}_B = W^T S_B W$$

Recall that we are looking for a projection that
- **maximizes the ration of between-class to within-class scatter.**

Since the projection is no longer a scalar (it has $C-1$ dimensions), we use the determinant of the scatter matrices to obtain a scalar objective function

$$J(W) = \frac{|\tilde{S}_B|}{|\tilde{S}_W|} = \frac{|W^T S_B W|}{|W^T S_W W|}$$

We will seek the projection matrix $W^*$ that maximizes this ratio

# LDA, C classes

- It can be shown that the optimal projection matrix $W^*$ is the one whose columns are the eigenvectors corresponding to the largest eigenvalues of the following generalized eigenvalue problem

$$W^* = [w_1^* | w_2^* | \dots | w_{C-1}^*] = \arg max \frac{|W^T S_B W|}{|W^T S_W W|} \quad \Rightarrow \quad (\mathrm{S_B} - \lambda_\mathrm{i} \mathrm{S_W}) \mathrm{w_i^*} = 0$$

**Note that**

$\mathrm{S_B}$ is the sum of $C$ matrices of rank $\leq 1$ and the mean vectors and constrained by

$$\frac{1}{C} \sum_{i=1}^{C} \mu_i = \mu$$

- therefore, $\mathrm{S_B}$ will be of rank $(C-1)$ or less

- This means that only $(C-1)$ of the eigenvalues $\lambda_i$ will be non-zero

The projections with maximum class seprability information are the eigenvectors corresponding to the largest eigenvalues of $S_W^{-1} S_B$

# Example, Iris

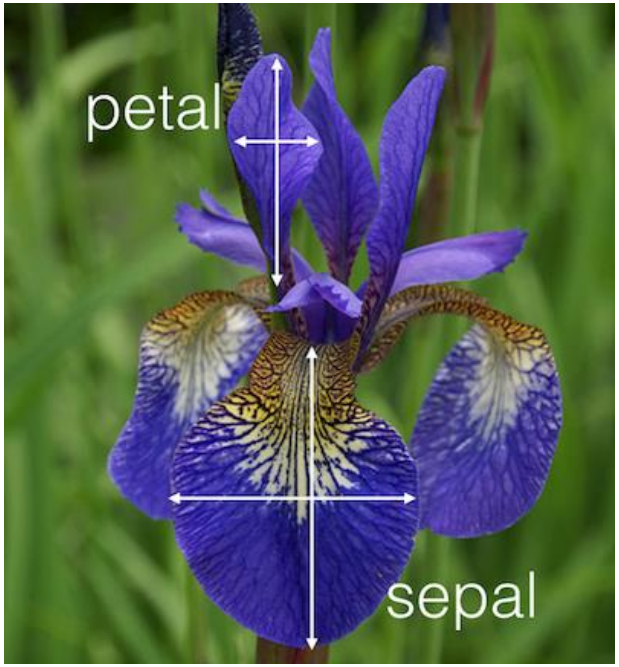
The iris dataset contains measurements for 150 iris flowers from three different species.

The three classes in the Iris dataset:

Iris-setosa (n=50)
Iris-versicolor (n=50)
Iris-virginica (n=50)

The four features of the Iris dataset:

sepal length in cm
sepal width in cm
petal length in cm
petal width in cm



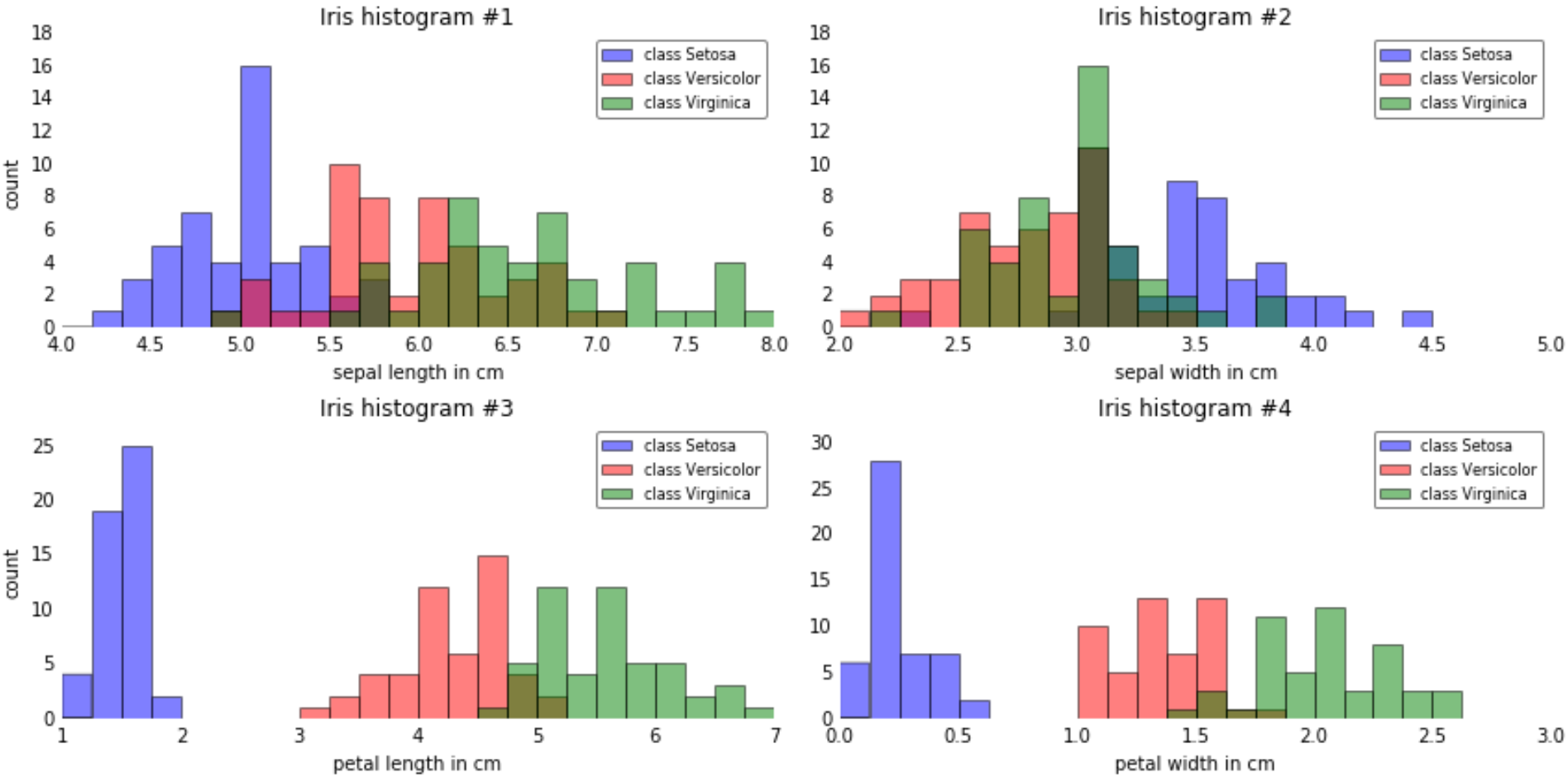
# Example, Iris

| | sepal length in cm | sepal width in cm | petal length in cm | petal width in cm | class label |
|---|---|---|---|---|---|
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

$$\boldsymbol{X} = \begin{bmatrix} x_{1_{\text{sepal length}}} & x_{1_{\text{sepal width}}} & x_{1_{\text{petal length}}} & x_{1_{\text{petal width}}} \\ x_{2_{\text{sepal length}}} & x_{2_{\text{sepal width}}} & x_{2_{\text{petal length}}} & x_{2_{\text{petal width}}} \\ \cdots & & & \\ x_{150_{\text{sepal length}}} & x_{150_{\text{sepal width}}} & x_{150_{\text{petal length}}} & x_{150_{\text{petal width}}} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \omega_{\text{setosa}} \\ \omega_{\text{setosa}} \\ \cdots \\ \omega_{\text{virginica}} \end{bmatrix}$$

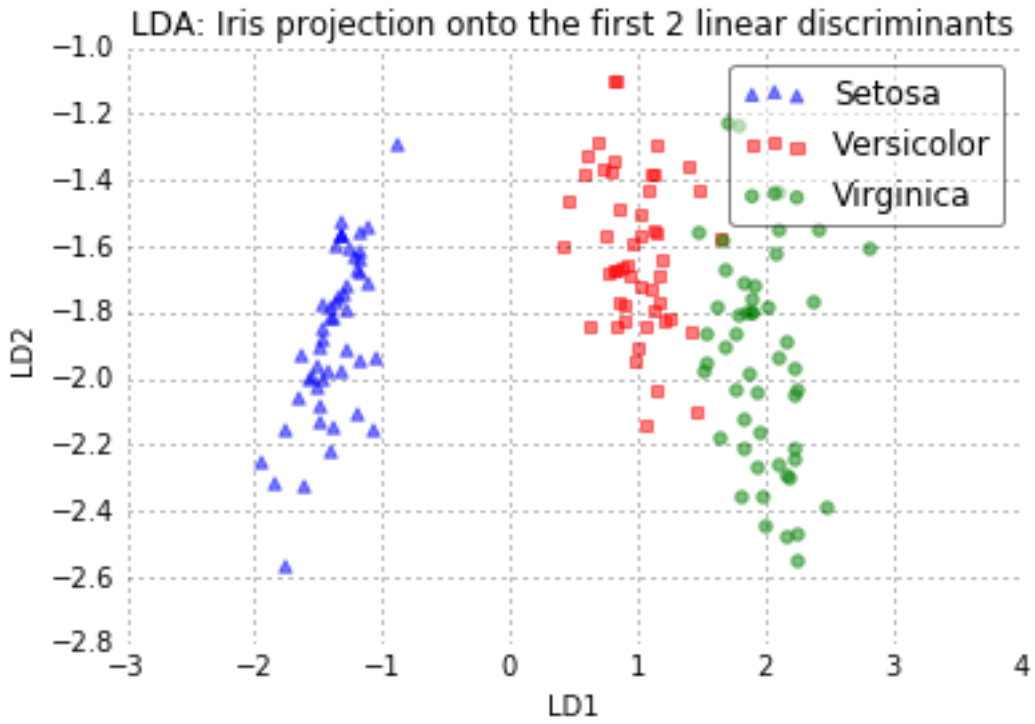
# Example, Iris

## Feature Selection



# Example, Iris

## Feature Extraction



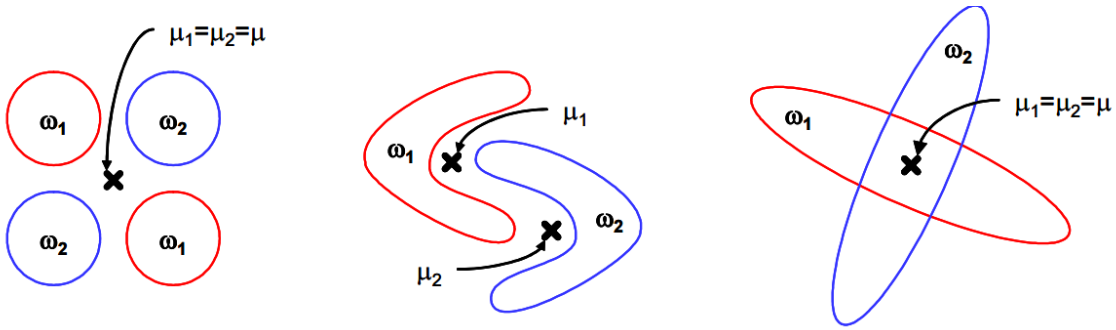
# Limitations of LDA

LDA produces **at most** $(\boldsymbol{C}-\mathbf{1})$ **feature projections**

If the classification error estimates establish that more features are needed, some other method must be employed to provide those additional features

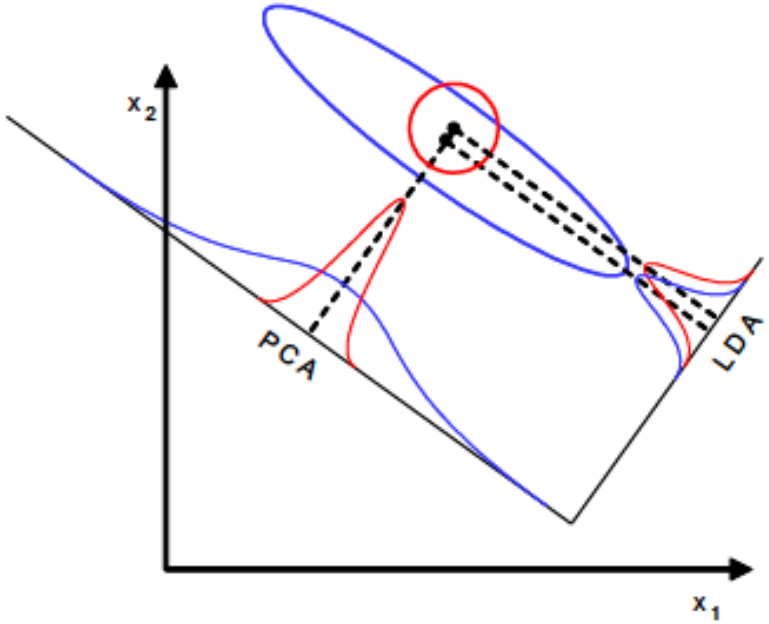
LDA is a parametric method ( **it assumes unimodal Gaussian likelihoods**)

- If the distributions are significantly non-Gaussian, the LDA projections may not preserve complex structure in the data needed for classification



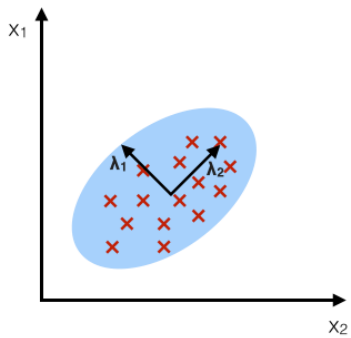
# Limitations of LDA

LDA will also fail if discriminatory information is not in the mean but in the variance of the data

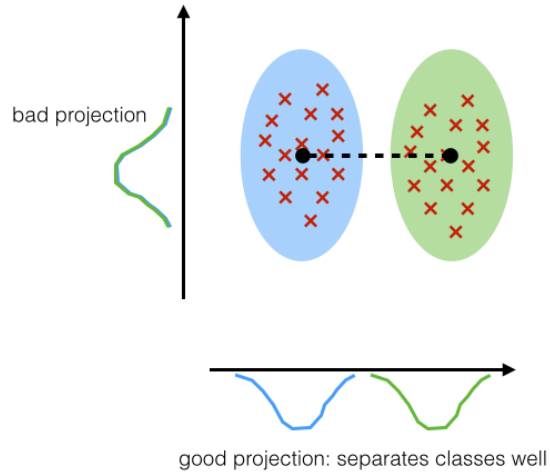


# PCA vs. LDA

**PCA:**
component axes that maximize the variance




**LDA:**
maximizing the component axes for class-separation




Both Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) are linear transformation techniques that are commonly used for dimensionality reduction.

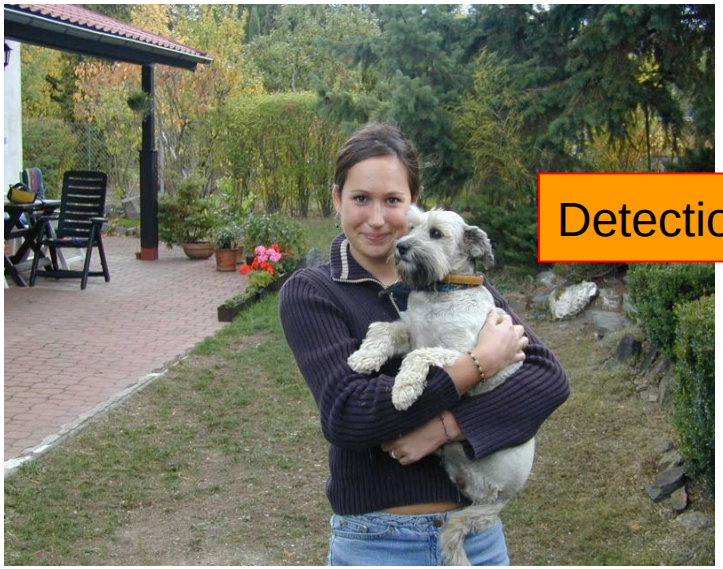
PCA can be described as an “unsupervised” algorithm, since it “ignores” class labels and its goal is to **find the directions (the so-called principal components) that maximize the variance in a dataset.**

LDA is “supervised” and computes **the directions (“linear discriminants”) that will represent the axes that that maximize the separation between multiple classes.**

# Real example

# Application of PCA and LDA
# - Face Recognition

# Face detection and recognition



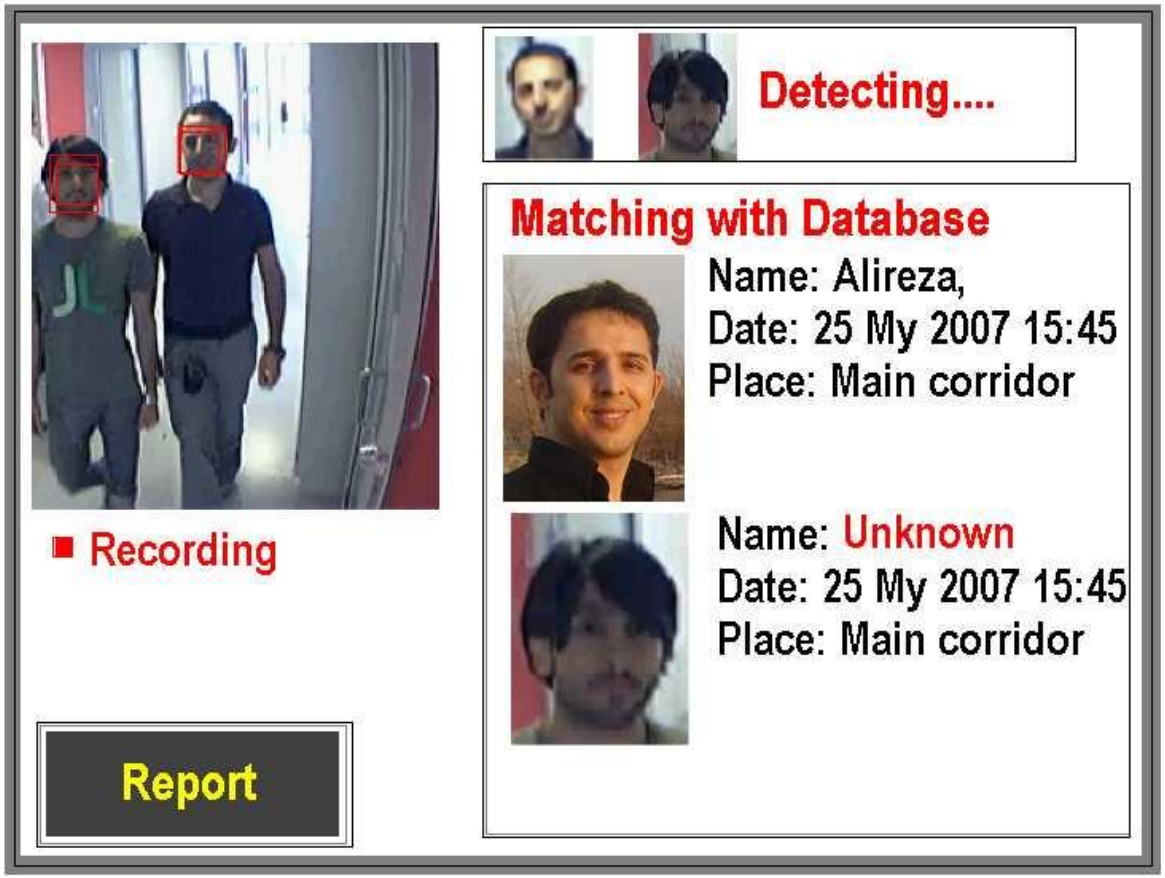
Detection

Recognition

“Sally”

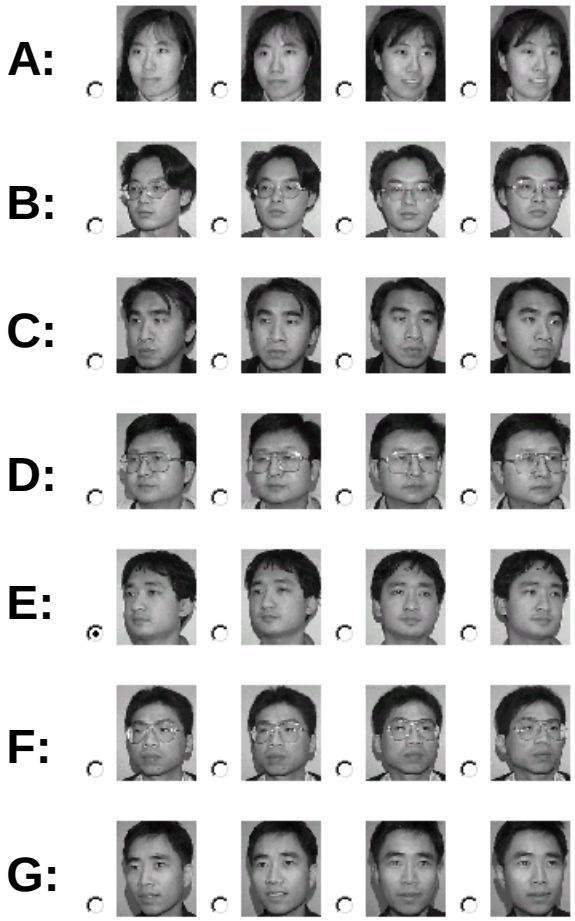
# Face detection and recognition

- Digital photography
- Surveillance



# Face Recognition

**Image database:**




**Test image:**

**Who is this guy?**

**Characteristics of FR:**

- **A mode of biometric identification**
- **Easy for human, hard for machine**

Input

- A dataset of face images of *n* person
- An unknown person's face image

Output:

Determine the identity of the unknown person

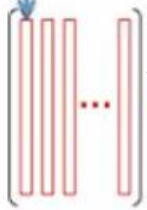
# Face Recognition

input: dataset of N face images

face: K x K bitmap of pixels

"unfold" each bitmap to $K^2$-dimensional vector

arrange in a matrix
each face = column

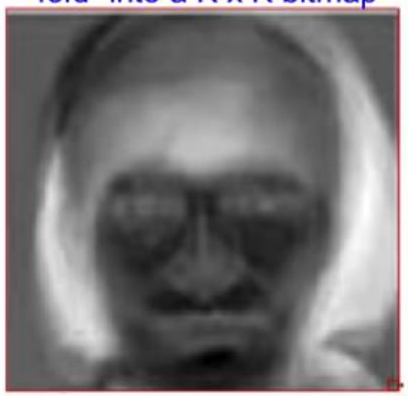


$X$

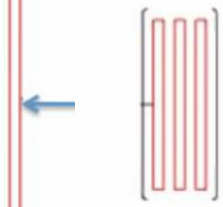
$K^2 \times N$

can visualize eigenvectors: *m* "aspects" of prototypical facial features

"fold" into a K x K bitmap



PCA

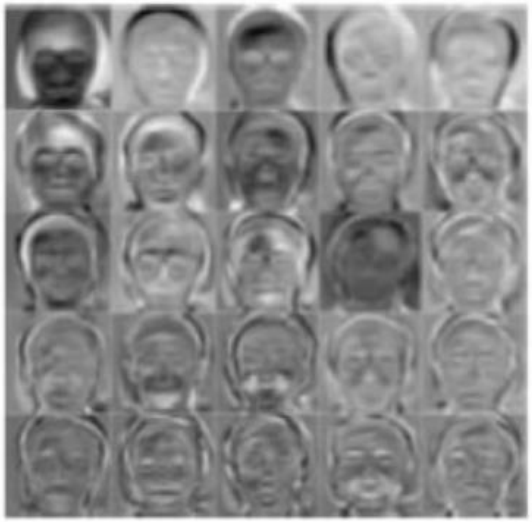


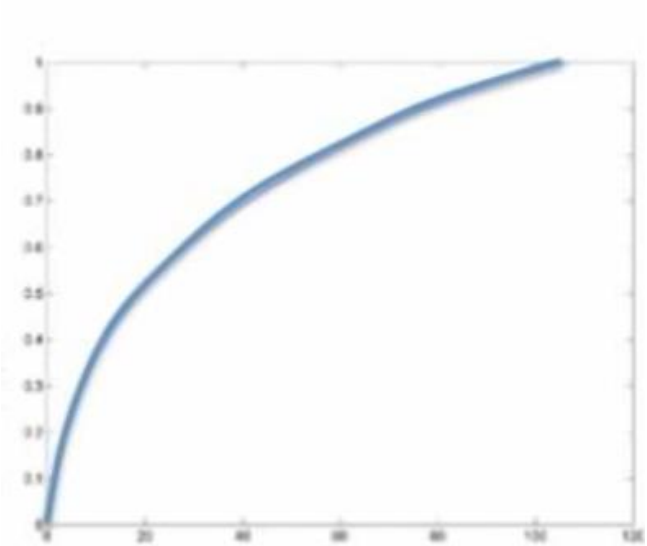
$K^2 \times m$

set of *m* eigenvectors
each is $K^2$-dimensional

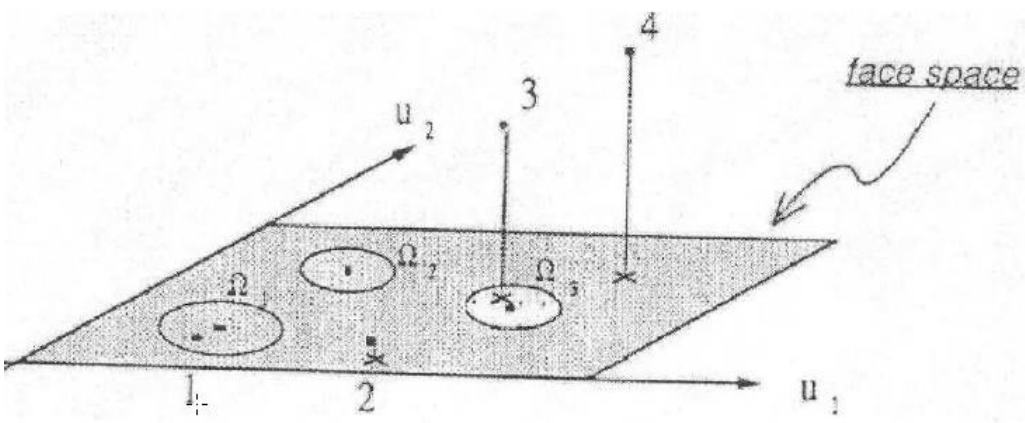
# Eigen Faces: Projection

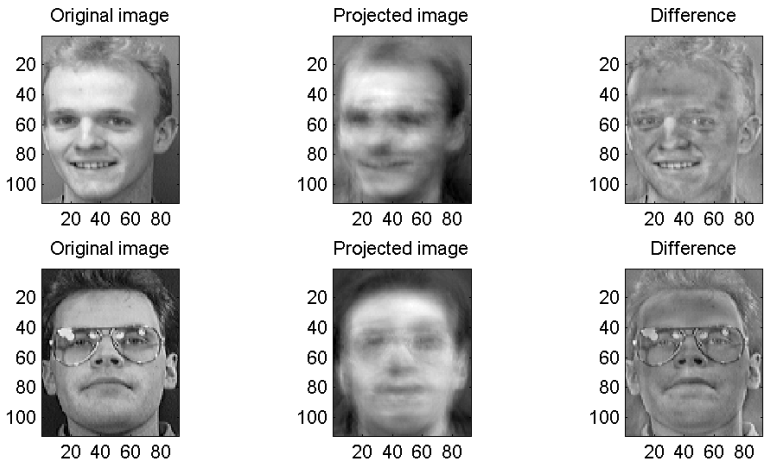

- Project new face to space of eigen-faces
- Represent vector as a linear combination of PCs
- How many do we need?





# Face Recognition




Given an unknown image $x$

- Step1: compute $\phi = x - \mu$
- Step2: compute $\hat{\phi} = \sum_{i=1}^{K} w_i u_i \quad (w_i = u_i^T \phi)$
- Step3: compute $e_d = ||\phi - \hat{\phi} \ \ ||$
- Step4: if $e_d < T_d$ , then $x$ is a face

# Steps of PCA

(Step 1) **Mean centering**

Compute the average $\bar{x}$ of given data $\{x_1, x_2, \ldots, x_m\}$, subtracted from the data

(Step 2) **Calculation of Covariance matrix($\mathbf{S} = \frac{1}{\mathbf{m}}\mathbf{X}\mathbf{X}^{\mathbf{T}}$) of data**

**For symmetric matrices, their eigenvectors always are at right angles to each other**

**Eigenvectors of covariance matrix are orthogonal**

(Step 3) **Calculation of eigenvalues and eigenvectors of covariance matrix (S)**

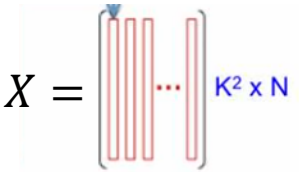
**Eigenvectors**: the **orientations** of the principal axes of the ellipse

**Eigenvalues**: the **lengths** of each of the principal axes of ellipse

(Step 4) **Mathematical representations of transformation of axes**

# Details for Step 3: PCA

Example. $K = 100$ and $N = 50$

$X =$ 

is 10,000x50 size matrix.

$S = \frac{1}{50} X X^T$ is a 10,000 x 10,000 size matrix.

**Problem:** $S$ is a large size matrix(10,000 x 10,000).
How to compute the eigenvectors of $S$?

Observation:

- $X^T X$ is a 50 x 50 size matrix

If $u$ is the eigenvector of $X^T X$, then $Xu$ is the eigenvector of $XX^T$.

If $\lambda$ is the eigenvalue of $X^T X$, then $\lambda$ is also the eigenvalue of $XX^T$.

$$(X^T X)\, u = \lambda u$$

$$(XX^T)(X\, u) = X(X^T X)\, u = X(\lambda u) = \lambda(Xu)$$

Note that:

$XX^T$ has 10,000 eigenvalues.
$X^T X$ has 50 eigenvalues, corresponding to the 50 largest eigenvalues of $XX^T$.

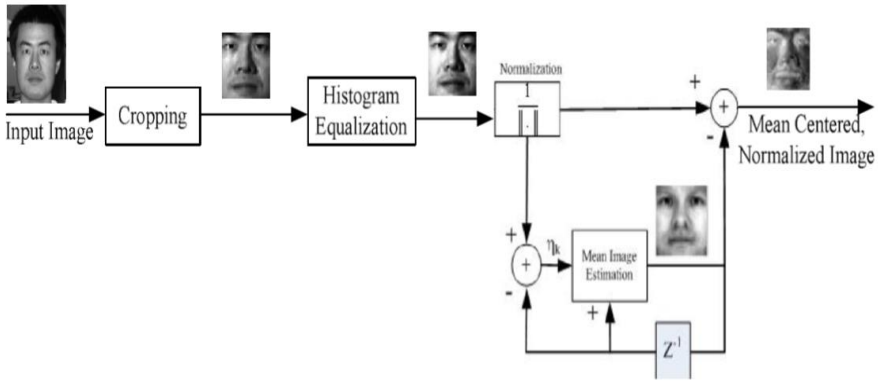
World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:5, 2008

# A New Face Recognition Method using PCA, LDA and Neural Network

A. Hossein Sahoolizadeh, B. Zargham Heidari, and C. Hamid Dehghani

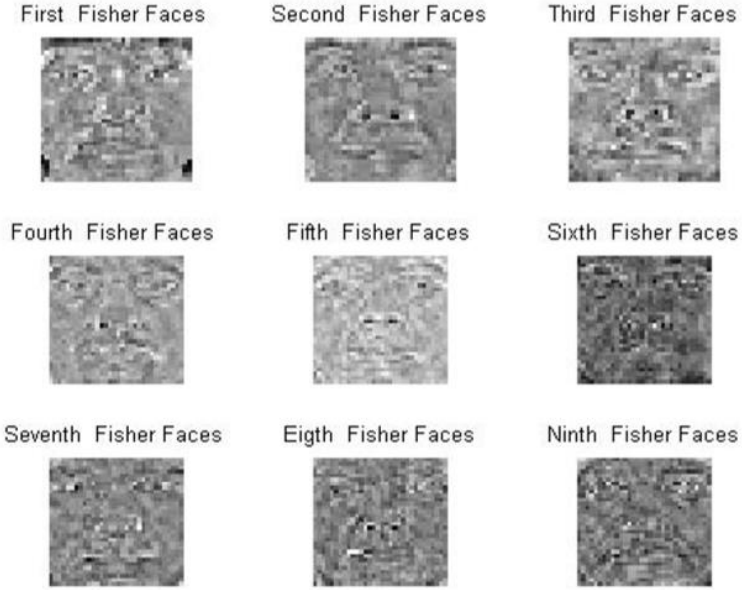
Input image (10 classes)

Mean centered data



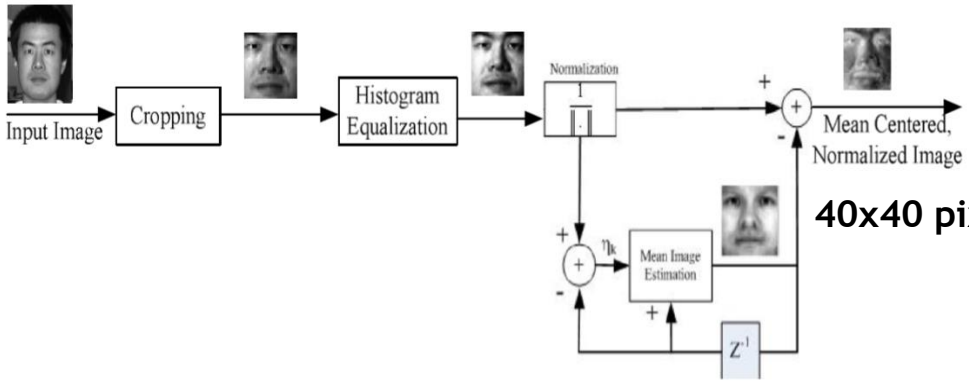

Eigen face via PCA

10 face images related to different classes

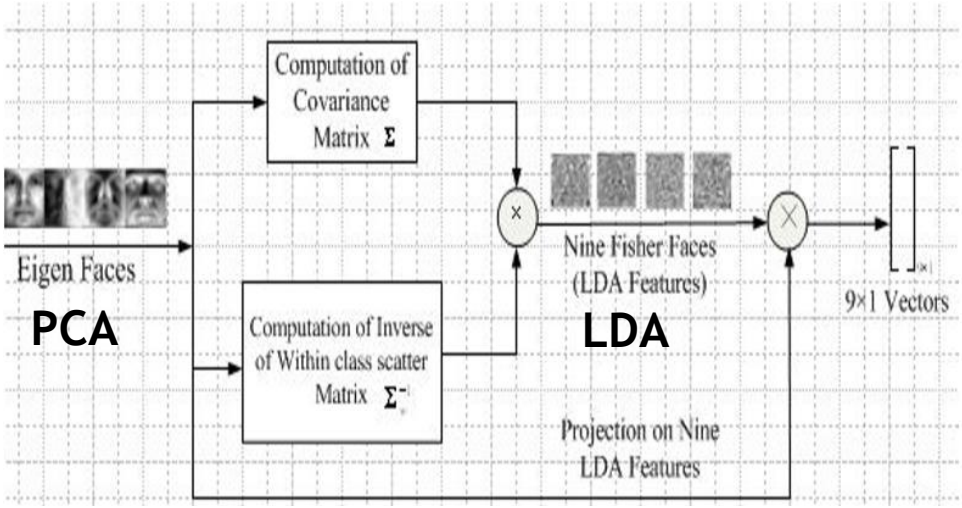


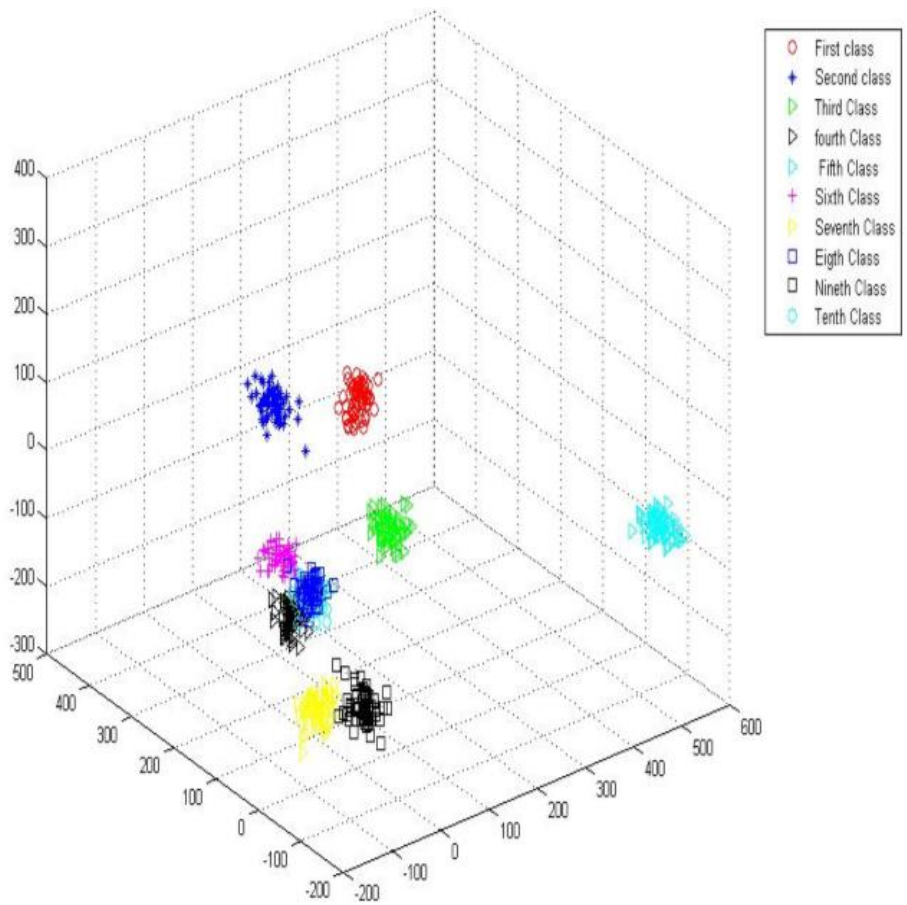

Fisher Faces via LDA

C-1 feature

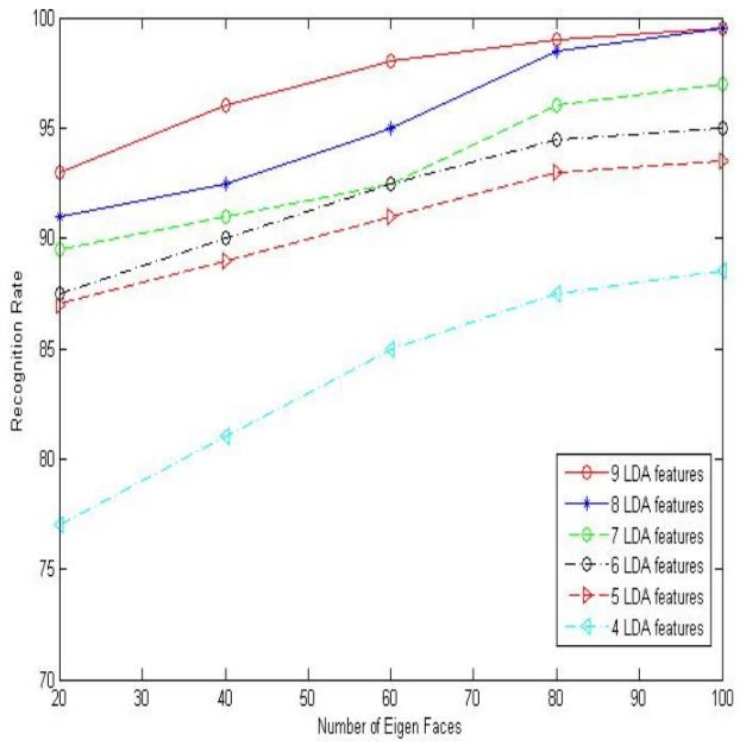



**40x40 pixels**






Distribution of classes in three dimensional LDA feature space. Axes are three most significant LDA features.




Comparison of recognition rate for choosing different values of PCA and LDA features

TABLE I.
Comparison of recognition rate by choosing different number of PCA and LDA features

| | 4 LDA features | 5 LDA features | 6 LDA features | 7 LDA features | 8 LDA features | 9 LDA features |
|---|---|---|---|---|---|---|
| 20 PCA features | 77% | 84% | 87.5 | 87.5% | 92% | 94.5 |
| 40 PCA features | 81% | 86% | 89.5% | 90% | 93.5% | 96.5% |
| 60 PCA features | 85% | 90.5% | 93% | 93% | 95% | 98% |
| 80 PCA features | 87.5 | 92% | 94.5 | 96 | 98.5% | 99% |
| 100 PCA features | 88.5% | 92.5% | 95% | 97% | 99.5% | 99.5% |

# Question??

Consumer application: iPhoto 2009

# Matrix Decomposition

## Dictionary Learning and Sparse Coding

## Nonnegative Matrix Factorization