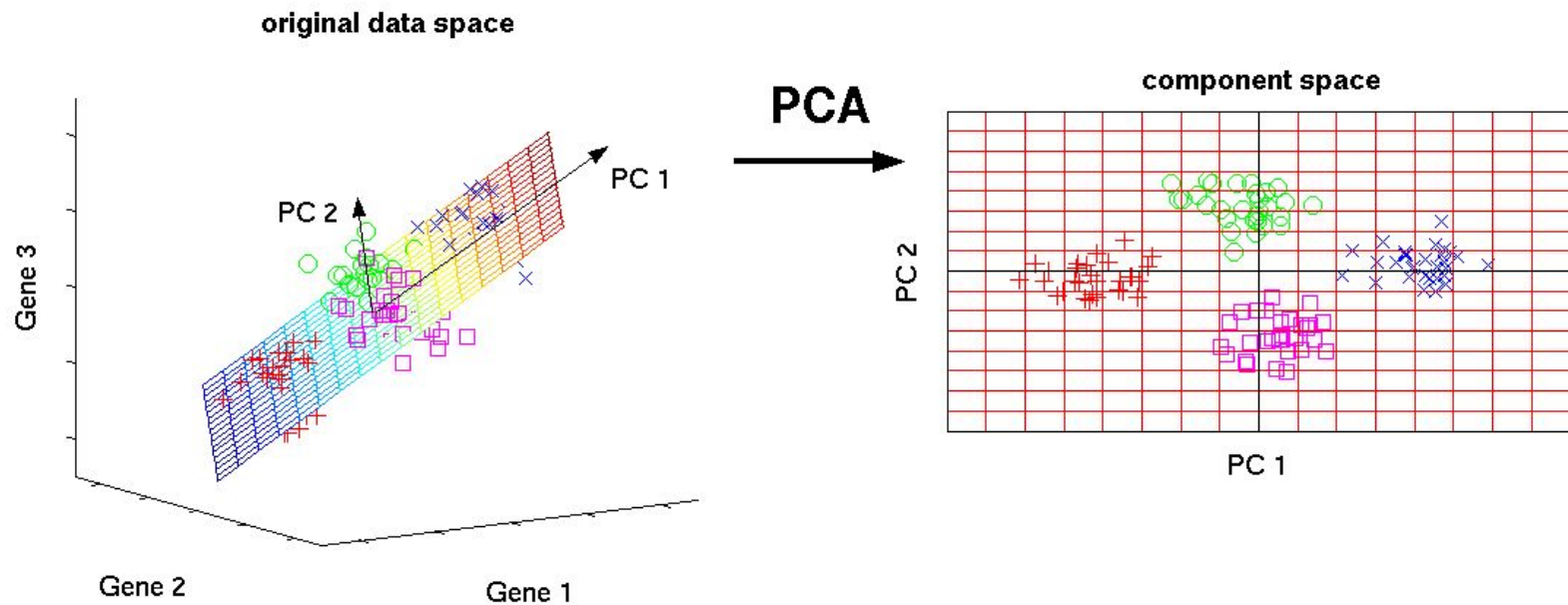


## Lecture 10

# PCA x LDA

<https://github.com/dalcimar/MA28CP-Intro-to-Machine-Learning>

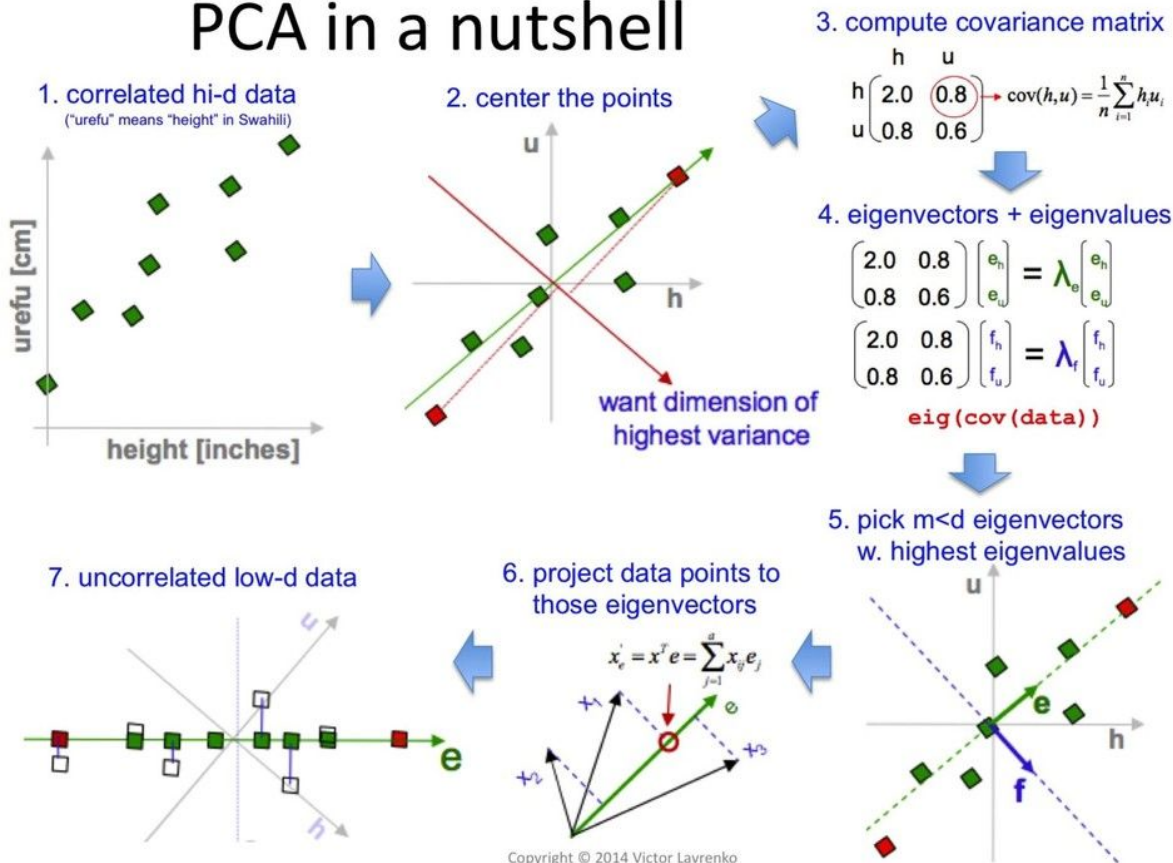
# Dimensionality reduction



# Dimensionality reduction

- O sistema de variabilidade do vetor aleatório composto das **p-variáveis originais** é **aproximado** pelo sistema de variabilidade do vetor aleatório que contém as **k-componentes** principais.
- A **qualidade** da aproximação **depende** do **número de componentes** mantidas no sistema e pode ser medida através da avaliação da proporção de variância total explicada por essas.

## PCA in a nutshell



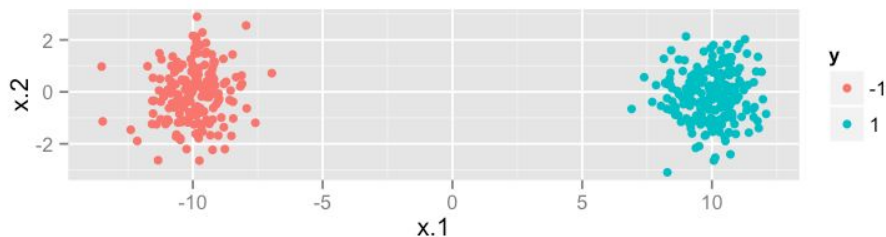
# PCA can worsen results of a classifier!

- Suppose you only have two (scaled and demeaned) features, denote them  $\mathbf{x}_1$  and  $\mathbf{x}_2$  with **positive correlation equal to 0.5, aligned in  $\mathbf{X}$** , and a third response variable  $Y$  you wish to classify.
- Suppose that the classification of  $Y$  is fully determined by the sign of  $\mathbf{x}_1$  and  $\mathbf{x}_2$ 
  - Performing PCA on  $\mathbf{X}$  results in the new (ordered by variance) features
  - Therefore, if you reduce your dimension to 1 i.e. the first principal component, you are throwing away the exact solution to your classification!

# PCA can worsen results of a classifier!

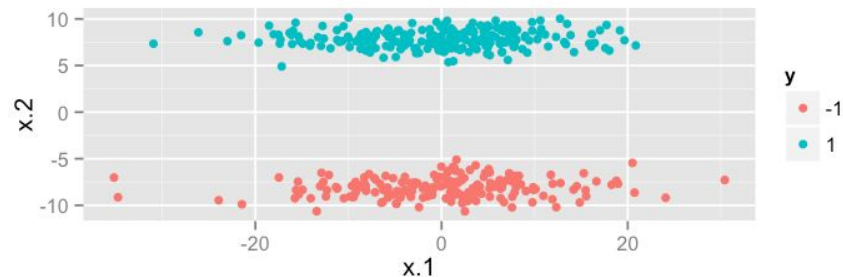
## PCA Helps

- The direction of maximal variance is horizontal, and the classes are separated horizontally.



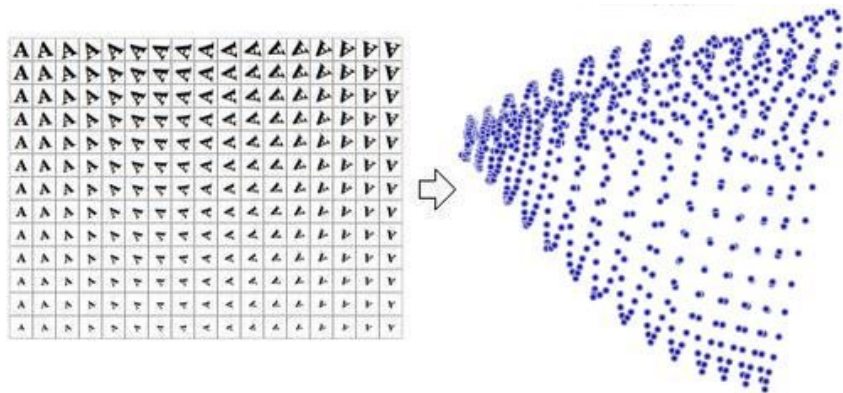
## PCA Hurts

- The direction of maximal variance is horizontal, but the classes are separated vertically

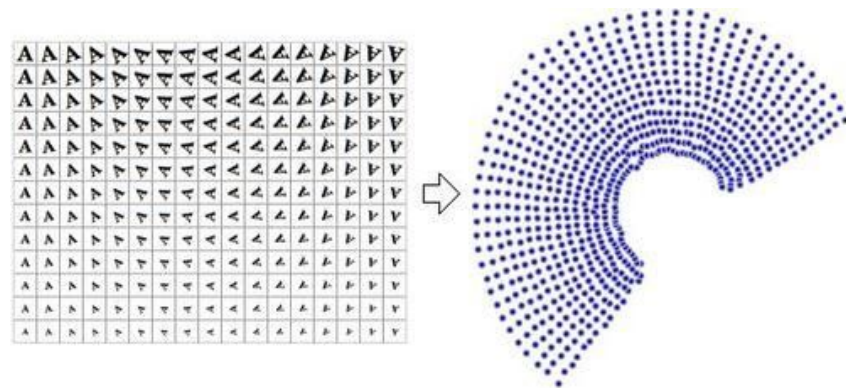


# PCA can worsen results of a classifier!

PCA is linear, It hurts when you want to see non linear dependencies.



A non linear algorithm (NLDR) with reduced images to 2 dimensions, rotation and scale:



# PCA can worsen results of a classifier!

Principal Component Analysis (PCA) is based on **extracting the axes on which data shows the highest variability**

- Although PCA “spreads out” data in the new basis, and can be of great help in unsupervised learning, **there is no guarantee that the new axes are consistent with the discriminatory features** in a (supervised) classification problem

# LDA x PCA

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

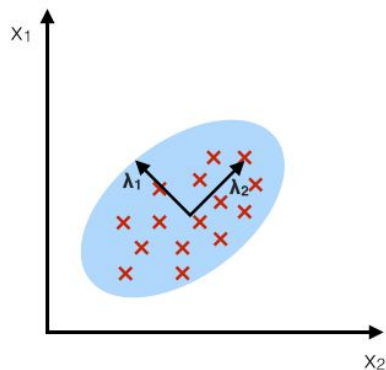
- **PCA can be described as an “unsupervised”** algorithm, since it **“ignores” class labels** and its goal is to find the directions (the so-called principal components) that **maximize the variance in a dataset**
- **LDA is “supervised”** and computes the directions (“linear discriminants”) that will represent the axes that **maximize the separation between multiple classes**



# LDA x PCA

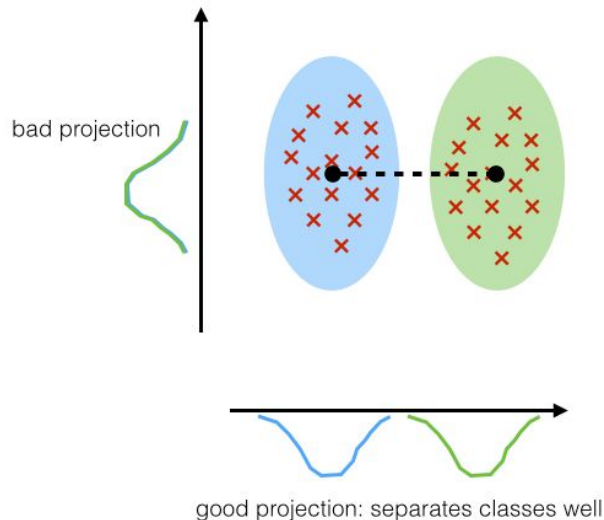
## PCA:

component axes that maximize the variance

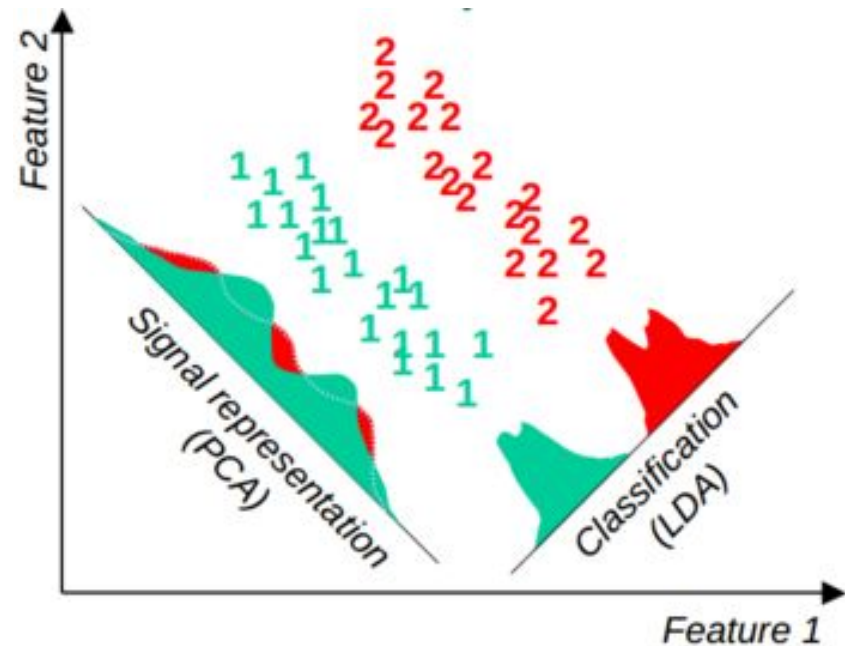
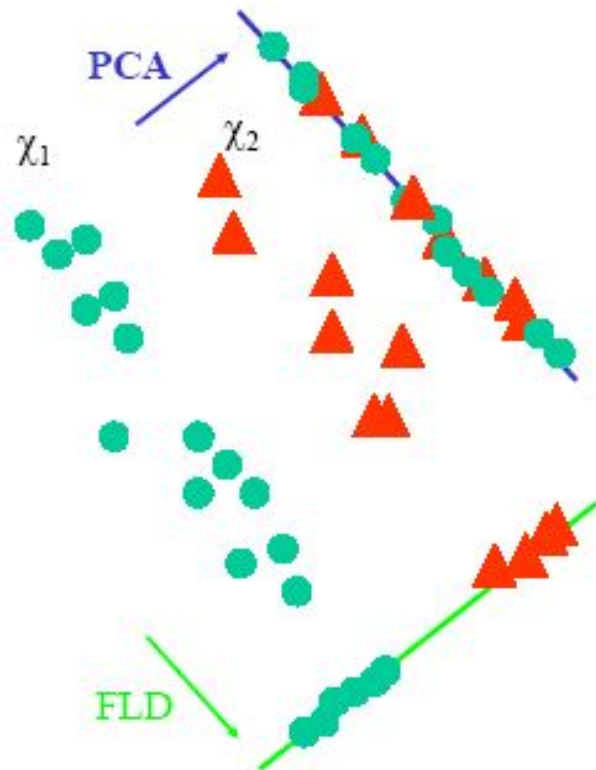


## LDA:

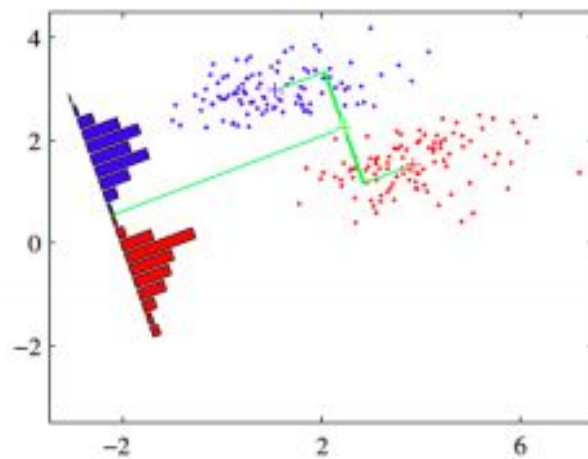
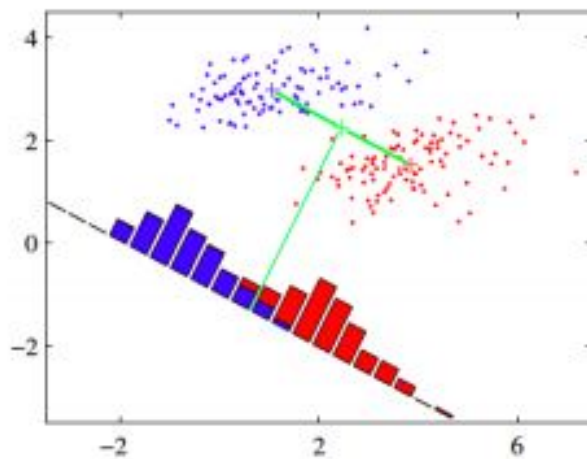
maximizing the component axes for class-separation



# LDA x PCA



# LDA x PCA



# Summarizing the LDA approach in 5 steps

1. ☐ Compute the d-dimensional mean vectors for the different classes from the dataset.
2. Compute the scatter matrices (in-between-class and within-class scatter matrix).
3. Compute the eigenvectors ( $e_1, e_2, \dots, e_d$ ) and corresponding eigenvalues ( $\lambda_1, \lambda_2, \dots, \lambda_d$ ) for the scatter matrices  $S_{\text{intra}}^{-1} * S_{\text{inter}}$ .
4. Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a  $d \times k$ -dimensional matrix  $W$  (where every column represents an eigenvector).
5. Use this  $d \times k$  eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the mathematical equation:  $y = W^T x$ 
  1. where  $x$  is a  $d \times 1$ -dimensional vector representing one sample, and  $y$  is the transformed  $k \times 1$ -dimensional sample in the new subspace.

# Scatter matrices

- A matriz de dispersão total (total scatter matrix) é definida por:

$$S = \sum_{i=1}^N (\vec{f}_i - \vec{M})(\vec{f}_i - \vec{M})^T$$

- A matriz de dispersão para a classe  $C_i$ , representada por  $S_i$ , indicando a dispersão de cada classe é dada por:

$$S_i = \sum_{i \in C_i} (\vec{f}_i - \vec{\mu}_i)(\vec{f}_i - \vec{\mu}_i)^T$$

# Scatter matrices

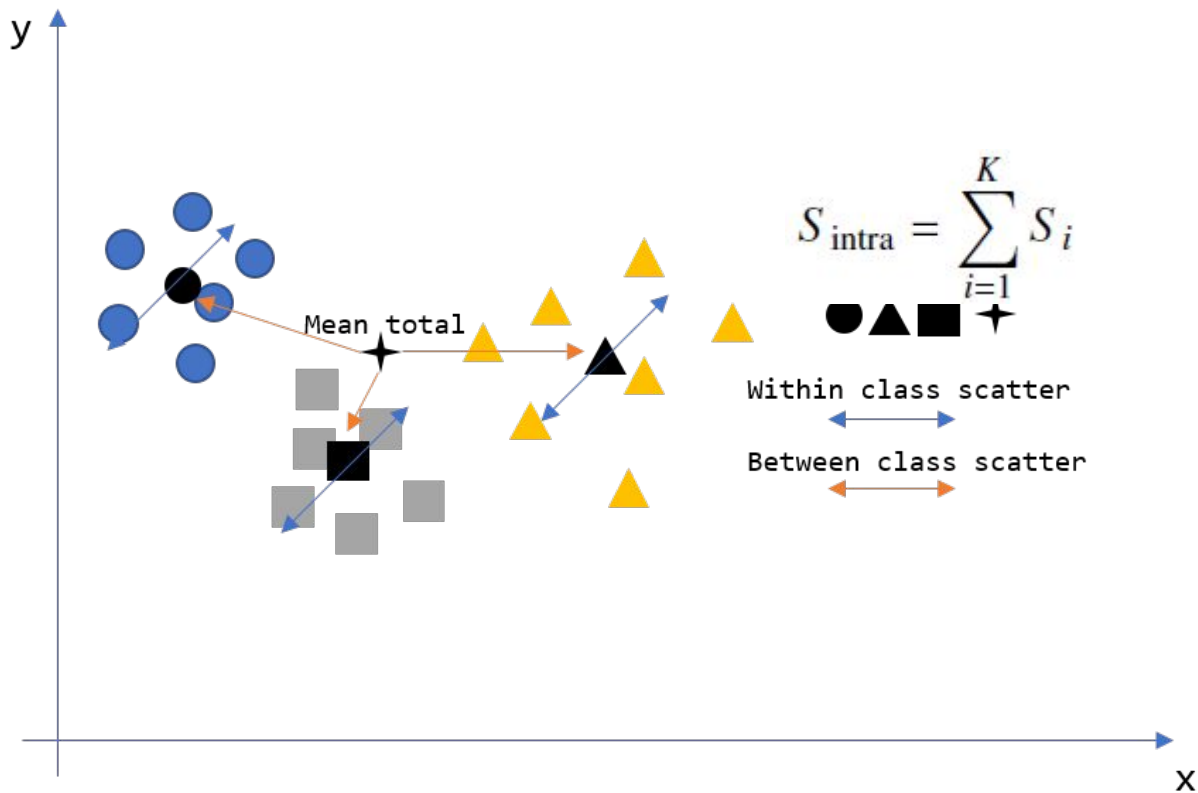
- A matrix de dispersão intra-classe é dada por:
  - indica a dispersão combinada de cada classe

$$S_{\text{intra}} = \sum_{i=1}^K S_i$$

- A matrix de dispersão inter-classes é dada por:
  - indica a dispersão das classes em termos de seus centróides




$$S_{\text{inter}} = \sum_{i=1}^K N_i (\vec{\mu}_i - \vec{M})(\vec{\mu}_i - \vec{M})^T$$

# Scatter matrices



$$S_{\text{inter}} = \sum_{i=1}^K N_i (\vec{\mu}_i - \vec{M}) (\vec{\mu}_i - \vec{M})^T$$

$$S_{\text{intra}} = \sum_{i=1}^K S_i$$

  
 Within class scatter  
  
 Between class scatter  


$$S_i = \sum_{i \in C_i} (\vec{f}_i - \vec{\mu}_i) (\vec{f}_i - \vec{\mu}_i)^T$$

$$S_{\text{intra}} = \sum_{i=1}^K S_i$$

# Scatter matrices

- Necessariamente temos:
  - a soma das matrizes intra e inter-classes é sempre preservada

$$S = S_{\text{intra}} + S_{\text{inter}}$$

- Essa conservação também pode ser provada por

$$\text{trace}(S) = \text{trace}(S_{\text{intra}}) + \text{trace}(S_{\text{inter}})$$



# Exemplo

- ▣  $N = 7$  objetos
- ▣  $K = 3$  classes
  - ▣  $C_1 = \text{contém } N_1 = 3 \text{ objetos}$
  - ▣  $C_2 = \text{contém } N_2 = 1 \text{ objetos}$
  - ▣  $C_3 = \text{contém } N_3 = 3 \text{ objetos}$
- ▣  $M = 2$  características

Object #	Class	Feature 1	Feature 2
1	$C_3$	9.2	33.2
2	$C_2$	5.3	21.4
3	$C_3$	8.8	31.9
4	$C_1$	2.9	12.7
5	$C_3$	9.0	32.4
6	$C_1$	1.5	12.0
7	$C_1$	1.2	11.5

# Exemplo

$$F = \begin{bmatrix} 9.2 & 33.2 \\ 5.3 & 21.4 \\ 8.8 & 31.9 \\ 2.9 & 12.7 \\ 9.0 & 32.4 \\ 1.5 & 12.0 \\ 1.2 & 11.5 \end{bmatrix}$$

$$\vec{M} = \begin{bmatrix} 5.4143 \\ 22.1571 \end{bmatrix}$$

Object #	Class	Feature 1	Feature 2
1	$C_3$	9.2	33.2
2	$C_2$	5.3	21.4
3	$C_3$	8.8	31.9
4	$C_1$	2.9	12.7
5	$C_3$	9.0	32.4
6	$C_1$	1.5	12.0
7	$C_1$	1.2	11.5

$$\vec{f}_1 = \begin{bmatrix} 9.2 \\ 33.2 \end{bmatrix}; \quad \vec{f}_2 = \begin{bmatrix} 5.3 \\ 21.4 \end{bmatrix}; \quad \vec{f}_3 = \begin{bmatrix} 8.8 \\ 31.9 \end{bmatrix}; \quad \vec{f}_4 = \begin{bmatrix} 2.9 \\ 12.7 \end{bmatrix};$$

$$\vec{f}_5 = \begin{bmatrix} 9.0 \\ 32.4 \end{bmatrix}; \quad \vec{f}_6 = \begin{bmatrix} 1.5 \\ 12.0 \end{bmatrix}; \quad \vec{f}_7 = \begin{bmatrix} 1.2 \\ 11.5 \end{bmatrix}$$

# Exemplo

Object #	Class	Feature 1	Feature 2
1	$C_3$	9.2	33.2
2	$C_2$	5.3	21.4
3	$C_3$	8.8	31.9
4	$C_1$	2.9	12.7
5	$C_3$	9.0	32.4
6	$C_1$	1.5	12.0
7	$C_1$	1.2	11.5

$$F_1 = \begin{bmatrix} 2.9 & 12.7 \\ 1.5 & 12.0 \\ 1.2 & 11.5 \end{bmatrix};$$

$$F_2 = \begin{bmatrix} 5.3 & 21.4 \end{bmatrix};$$

$$F_3 = \begin{bmatrix} 9.2 & 33.2 \\ 8.8 & 31.9 \\ 9.0 & 32.4 \end{bmatrix}.$$

$$\vec{\mu}_1 = \begin{bmatrix} 1.8667 \\ 12.0667 \end{bmatrix};$$

$$\vec{\mu}_2 = \begin{bmatrix} 5.3 \\ 21.4 \end{bmatrix};$$

$$\vec{\mu}_3 = \begin{bmatrix} 9.0 \\ 32.5 \end{bmatrix}$$

## Exemplo

$$\begin{aligned} S &= \sum_{i=1}^N (\vec{f}_i - \vec{M})(\vec{f}_i - \vec{M})^T \\ &= \begin{bmatrix} 9.2 - 5.4143 \\ 33.20 - 22.1571 \end{bmatrix} \begin{bmatrix} 9.2 - 5.4143 & 33.20 - 22.1571 \end{bmatrix} + \dots + \\ &\quad + \begin{bmatrix} 1.2 - 5.4143 \\ 11.5 - 22.1571 \end{bmatrix} \begin{bmatrix} 1.2 - 5.4143 & 11.5 - 22.1571 \end{bmatrix} \\ &= \begin{bmatrix} 78.0686 & 220.0543 \\ 220.0543 & 628.5371 \end{bmatrix}. \end{aligned}$$

# Exemplo

$$\begin{aligned} S_1 &= \sum_{i \in C_1} (\vec{f}_i - \vec{\mu}_1)(\vec{f}_i - \mu_1)^T \\ &= \begin{bmatrix} 2.9 - 1.8667 \\ 12.7 - 12.0667 \end{bmatrix} \begin{bmatrix} 2.9 - 1.8667 & 12.7 - 12.0667 \end{bmatrix} + \\ &\quad + \begin{bmatrix} 1.5 - 1.8667 \\ 12 - 12.0667 \end{bmatrix} \begin{bmatrix} 1.5 - 1.8667 & 12 - 12.0667 \end{bmatrix} + \\ &\quad + \begin{bmatrix} 1.2 - 1.8667 \\ 11.5 - 12.0667 \end{bmatrix} \begin{bmatrix} 1.2 - 1.8667 & 11.5 - 12.0667 \end{bmatrix} \\ &= \begin{bmatrix} 1.6467 & 1.0567 \\ 1.0567 & 0.7267 \end{bmatrix}. \end{aligned}$$

# Exemplo

$$\begin{aligned} S_2 &= \sum_{i \in C_2} (\vec{f}_i - \vec{\mu}_2) (\vec{f}_i - \mu_2)^T = \begin{bmatrix} 5.3 - 5.3 \\ 21.4 - 21.4 \end{bmatrix} \begin{bmatrix} 5.3 - 5.3 & 21.4 - 21.4 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \end{aligned}$$

$$\begin{aligned} S_3 &= \sum_{i \in C_3} (\vec{f}_i - \vec{\mu}_3) (\vec{f}_i - \mu_3)^T \\ &= \begin{bmatrix} 9.2 - 9 \\ 33.2 - 32.5 \end{bmatrix} \begin{bmatrix} 9.2 - 9 & 33.2 - 32.5 \end{bmatrix} + \begin{bmatrix} 8.8 - 9 \\ 31.9 - 32.5 \end{bmatrix} \begin{bmatrix} 8.8 - 9 & 31.9 - 32.5 \end{bmatrix} + \\ &\quad + \begin{bmatrix} 9 - 9 \\ 32.4 - 32.5 \end{bmatrix} \begin{bmatrix} 9 - 9 & 32.4 - 32.5 \end{bmatrix} \\ &= \begin{bmatrix} 0.08 & 0.26 \\ 0.26 & 0.86 \end{bmatrix}. \end{aligned}$$

# Exemplo

$$S_{\text{intra}} = \sum_{i=1}^K S_i = S_1 + S_2 + S_3 = \begin{bmatrix} 1.7267 & 1.3167 \\ 1.3167 & 1.5867 \end{bmatrix}$$

$$\begin{aligned} S_{\text{inter}} &= \sum_{i=1}^K N_i (\vec{\mu}_i - \vec{M})(\vec{\mu}_i - \vec{M})^T \\ &= (3) \begin{bmatrix} 1.8667 - 5.4143 \\ 12.0667 - 22.1571 \end{bmatrix} \begin{bmatrix} 1.8667 - 5.4143 & 12.0667 - 22.1571 \end{bmatrix} + \\ &\quad + (1) \begin{bmatrix} 5.3 - 5.4143 \\ 21.4 - 22.1571 \end{bmatrix} \begin{bmatrix} 5.3 - 5.4143 & 21.4 - 22.1571 \end{bmatrix} + \\ &\quad + (3) \begin{bmatrix} 9 - 5.4143 \\ 32.5 - 22.1571 \end{bmatrix} \begin{bmatrix} 9 - 5.4143 & 32.5 - 22.1571 \end{bmatrix} \\ &= \begin{bmatrix} 72.3419 & 218.7376 \\ 218.7376 & 626.9505 \end{bmatrix}. \end{aligned}$$

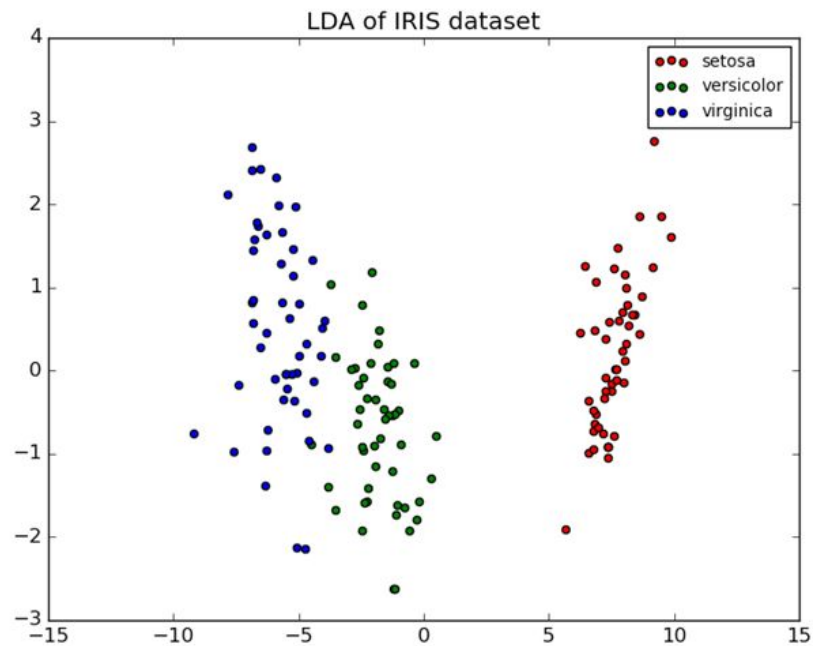
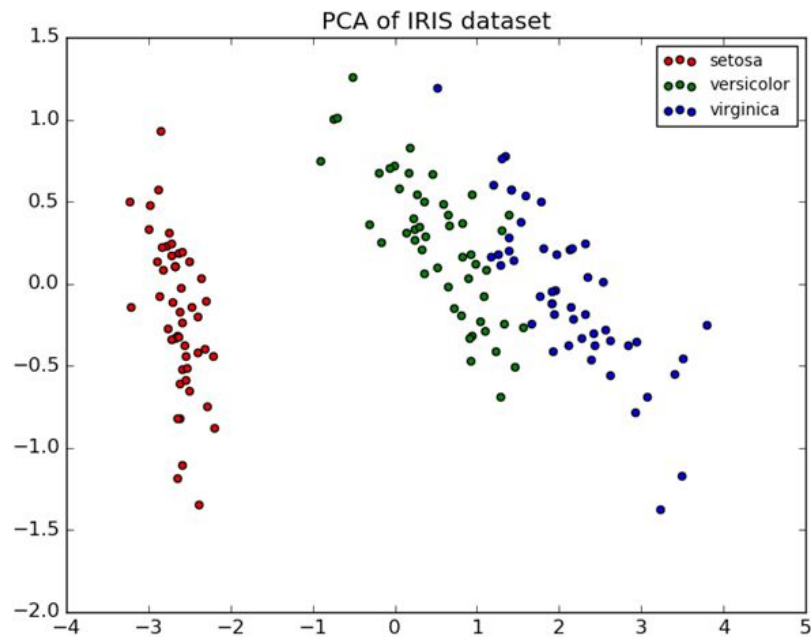
## Exemplo

$$\begin{aligned} S_{\text{intra}} + S_{\text{inter}} &= \begin{bmatrix} 1.7267 & 1.3167 \\ 1.3167 & 1.5867 \end{bmatrix} + \begin{bmatrix} 76.3419 & 218.7376 \\ 218.7376 & 626.9505 \end{bmatrix} \\ &\cong \begin{bmatrix} 78.0686 & 220.0543 \\ 220.0543 & 628.5371 \end{bmatrix} \\ &= S, \end{aligned}$$

$$\text{trace}(S_{\text{intra}}) + \text{trace}(S_{\text{inter}}) = 3.3133 + 703.2924 \cong 706.6057 = \text{trace}(S)$$



# LDA x PCA



# LDA x PCA

Although it might sound intuitive that LDA is superior to PCA for a multi-class classification task where the class labels are known, this might not always be the case.

- For example, comparisons between classification accuracies for image recognition after using PCA or LDA show that PCA tends to outperform LDA if the number of samples per class is relatively small
- In practice, it is also not uncommon to use both LDA and PCA in combination:
- E.g., PCA for dimensionality reduction followed by an LDA

# Summarizing

**PCA (Principal Component Analysis):** is unsupervised or what is the same, it does not use class-label information. Therefore, discriminative information is not necessarily preserve.

- Minimizes the projection error
- Maximizes the variance of projected points
- Example: Reducing the number of features of an face (face detection)

**LDA (Linear Discriminant Analysis):** A PCA that takes class-labels into consideration, hence, it's supervised.

- Maximizes distance between classes
- Minimizes distance within classes
- Example: Separating faces into male and female clusters (face recognition)

# Summarizing

- If we would observe that all eigenvalues **have a similar magnitude**, then this may be a **good indicator** that our data is already projected on a “**good**” **feature space**.
- And in the other scenario, if some of the **eigenvalues are much larger than others**, we might be **interested in keeping only those eigenvectors** with the highest eigenvalues, since they contain more information about our data distribution.
  - Vice versa, eigenvalues that are close to 0 are less informative and we might consider dropping those for constructing the new feature subspace.