3. Selección y Extracción de características

Selección:

- óptimos y subóptimos

Extracción:

- PCA
- LDA
- ICA
- NMF

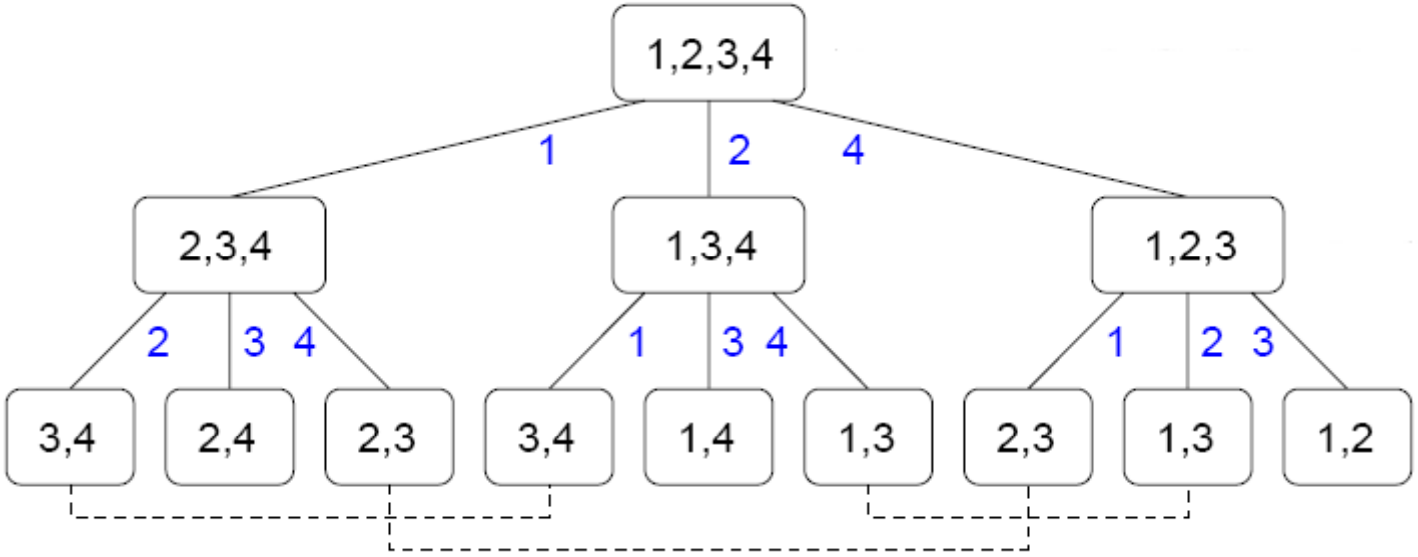
Selección de Características

- Objetivo: Seleccionar un conjunto de p variables a partir de un conjunto inicial de d variables, con la restricción de optimizar una función criterio (Separabilidad entre las clases)
- Ejemplo de Optimo: Ramificación y poda (Branch and Bound)
  - Proporcionan una solución óptima pero a costa de un gran esfuerzo computacional

Selección de Características

BRANCH AND BOUND

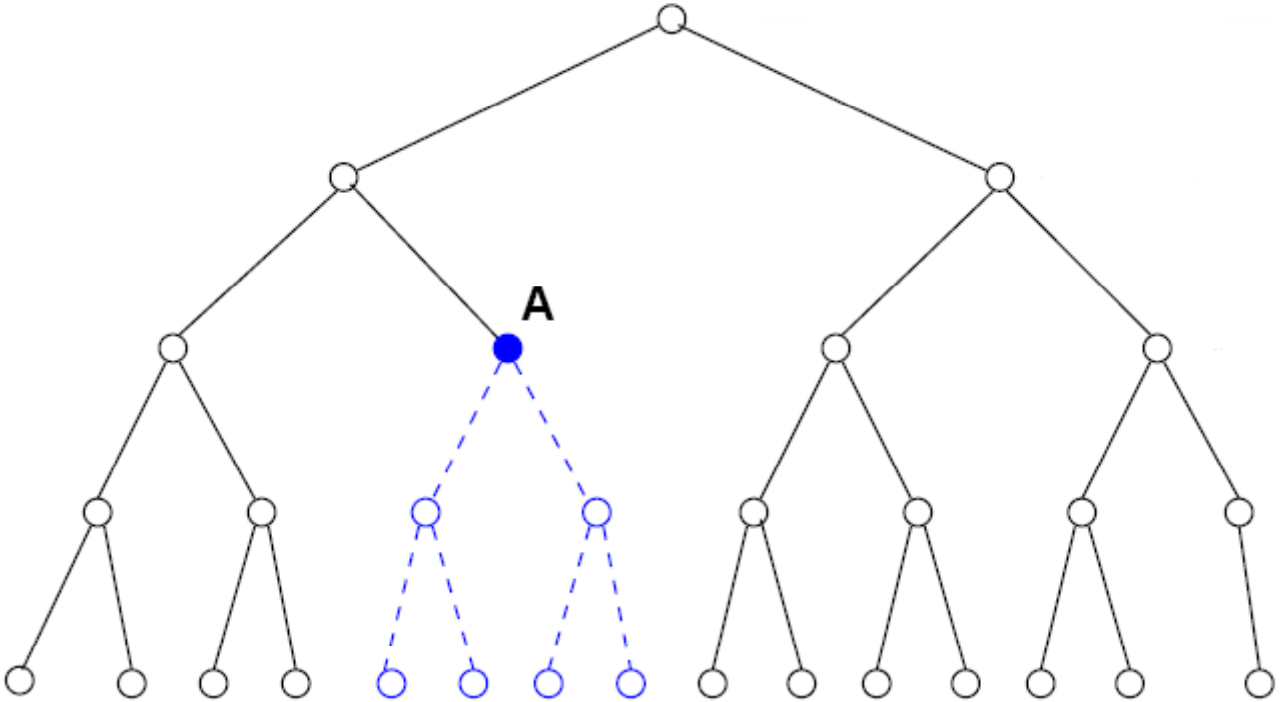
Construcción de un árbol (Ej. d=4, p=2)



Selección de Características

BRANCH AND BOUND

No es necesario construir el árbol al completo



Extracción de Características

- Objetivo: Transformar el espacio de patrones original P en un nuevo espacio P’ que no es necesariamente un subespacio de P. (Obtención de nuevos atributos con algún significado)

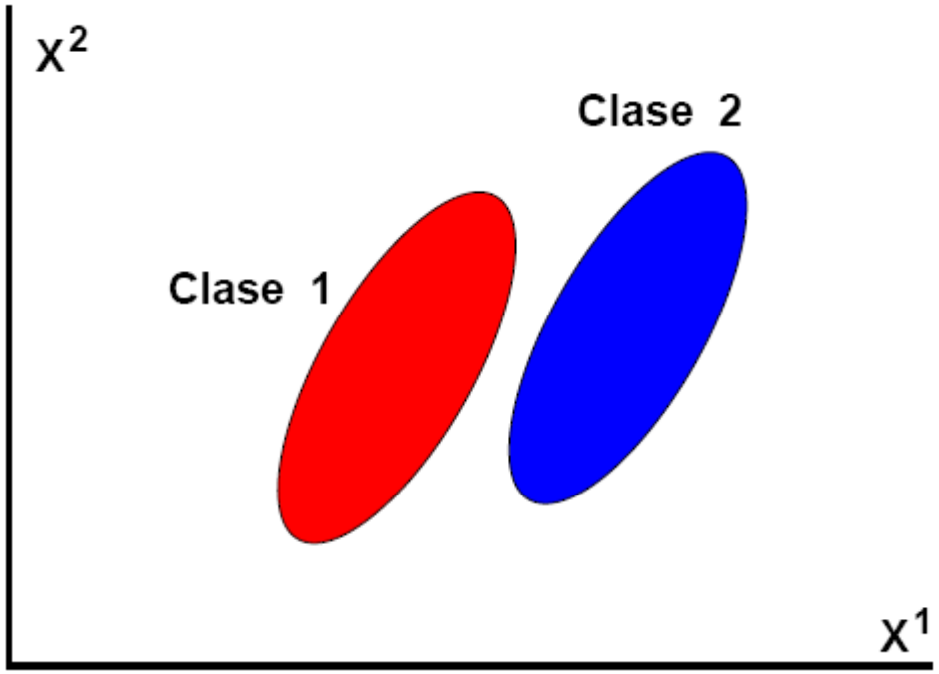
$$Y=g(X)$$

Considerando funciones lineales

$$Y=G\ X$$

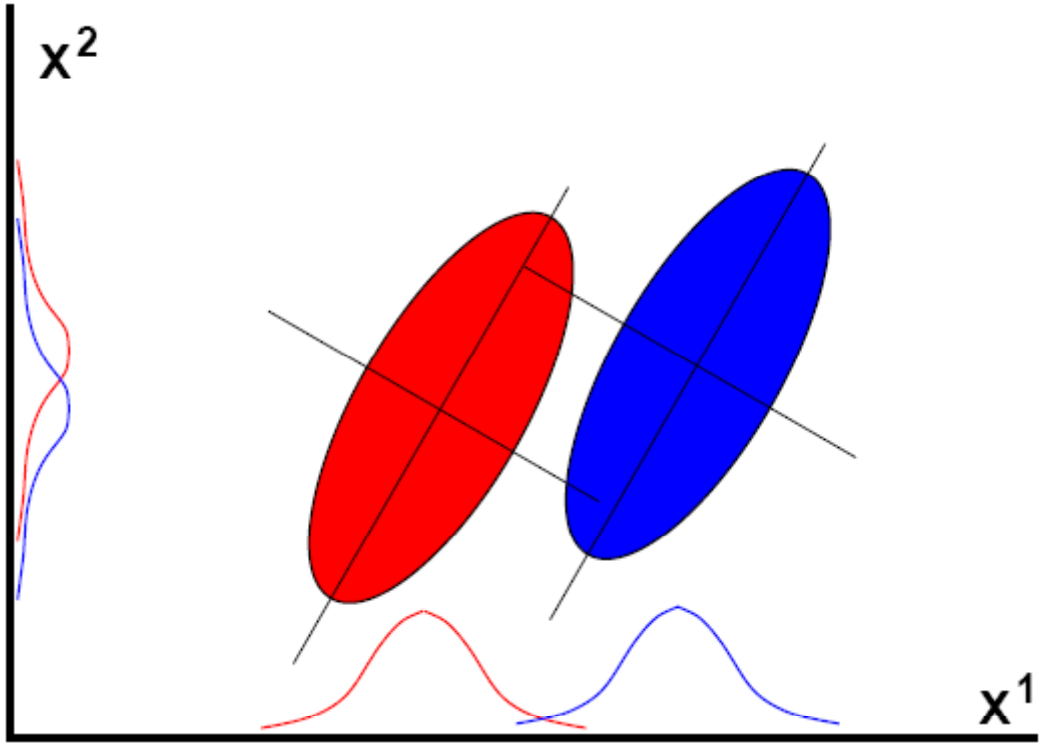
Extracción de Características

- Ejemplo de clasificación entre dos clases. El número de características iniciales es 2 (d=2: X^1, X^2). Se desea utilizar una única característica para separar estas dos clases (p=1)



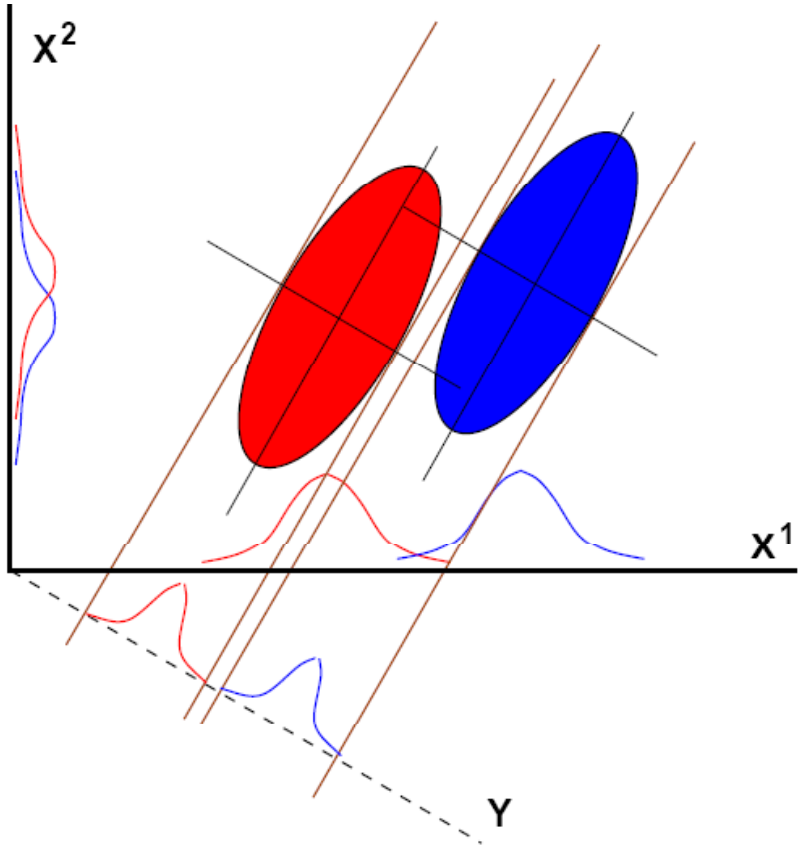
Extracción de Características

- La SELECCIÓN de cualquiera de las variables originales no proporciona una buena solución



Extracción de Características

- Si se usa una nueva variable Y de la forma $Y=-0.8X^1+0.5X^2$, permite obtener mejores resultados



Principal Component Analysis

- PCA = Karhunen-Loeve transform = Hotelling transform
- PCA es el método más popular de extracción de características
- PCA es una transformación lineal
- PCA se utiliza frecuentemente como base para el reconocimiento de caras basadas en apariencia

PCA (Principal Component Analysis)

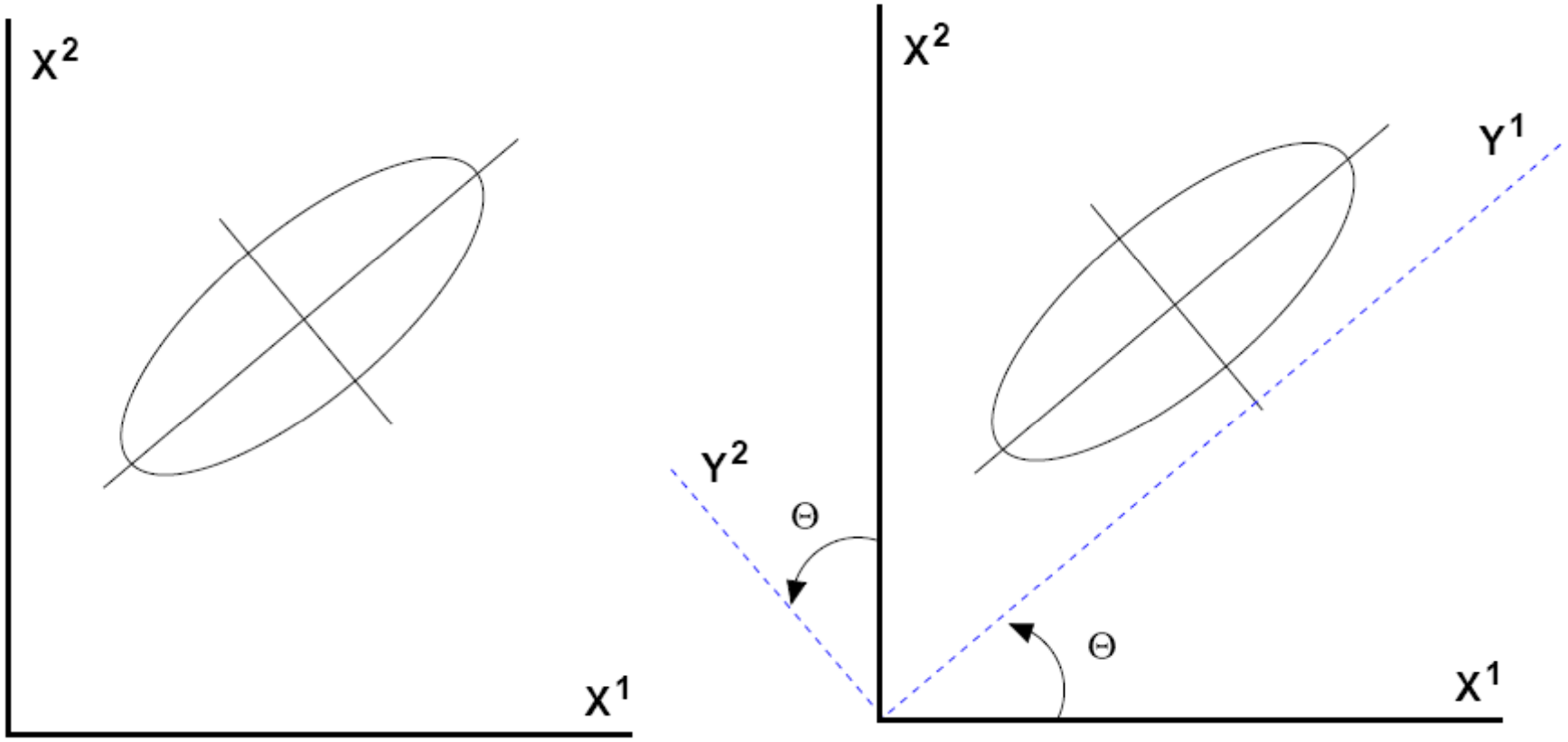
Objetivo:

Transformar el espacio de representación P en uno nuevo P’, en el que los datos estén incorrelados (la matriz de covarianza en P’ será diagonal).

Es decir, se trata de encontrar un nuevo conjunto de ejes ortogonales en el que la varianza de los datos sea la máxima en alguna dirección.

PCA (Principal Component Analysis)

- Ejemplo



PCA (Principal Component Analysis)

- Ejemplo

La relación entre los nuevos ejes y los antiguos consiste en:

$$Y_1=W_{11}X_1+W_{12}X_2$$
$$Y_2=W_{21}X_1+W_{22}X_2$$

La mayor parte de la información contenida en el espacio P puede retenerse únicamente en el primer eje principal Y^1, lo que implica una selección de características.

PCA (Principal Component Analysis)

Datos de mayor dimensión:

Los nuevos ejes se definen secuencialmente de manera que un nuevo eje se define como aquel que es perpendicular al seleccionado anteriormente y su dirección es la de la máxima varianza de entre todos los ejes posibles.

PCA (Principal Component Analysis)

Objetivo:

Encontrar W_{ij}

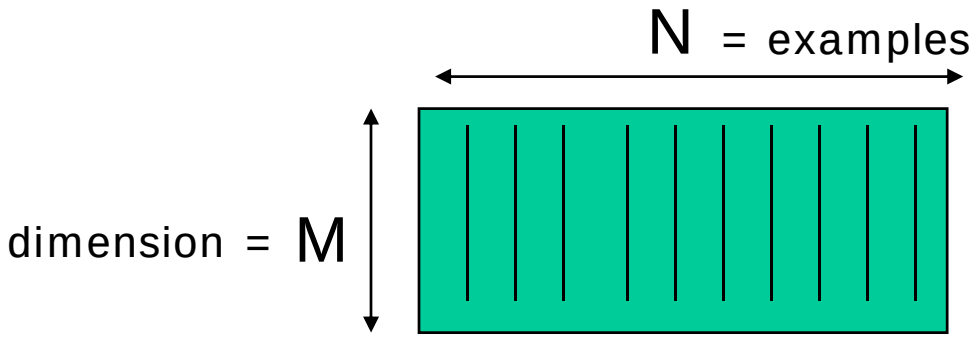
$$\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$$

1. Los ejes que definen P’ son ortogonales
2. Los datos en P’ están incorrelados

PCA – Maximización de Varianza

Espacios multidimensionales

Supongamos que tenemos **N** vectores de dimensión **M** ($\mathbf{x}_j$) alineados en una matriz de datos **X**



PCA - Maximización de Varianza

Sea **u** un vector director (vector de dimensión 1). La proyección del vector $\mathbf{x_j}$ sobre el vector **u** se puede calcular como:

$$p_j = \vec{u}^T \cdot \vec{x}_j = \sum_{i=1}^{M} u_i x_{ij}$$

Se desea encontrar la dirección **u** que maximiza la varianza de las proyecciones de todos los vectores $\mathbf{x_j}$ sobre esta dirección.

PCA - Maximización de Varianza

La función a maximizar por lo tanto es:

$$J^{PCA}(\vec{u}) = \sigma^2(p_j) = \frac{1}{N}\sum_{j=1}^{N}(p_j - \overline{p})^2 = \ldots = \vec{u}^T C\vec{u}$$

donde **C** es la matriz de covarianzas de la matriz de datos **X.**

Usando la técnica de Multiplicadores de Lagrange, la solución a este problema de maximización consiste en calcular los vectores propios y los valores propios de la matriz de covarianzas **C.**

PCA – Maximización de Varianza

El mayor valor propio se corresponde con el de máxima varianza, mientras que su correspondiente vector propio determina la dirección de esta máxima varianza.

Se puede diagonalizar la matriz **C**, mediante una descomposición de valores singulares (SVD) de la matriz de covarianzas **C:**

$$C = U \Lambda U^T$$

PCA – Maximización de Varianza

De tal forma que la matriz ortonormal **U** contiene los vectores propios $\mathbf{u_1}$, $\mathbf{u_2}$,.. $\mathbf{u_N}$ como columnas y la matriz diagonal Λ contiene los valores propios $\lambda_1, \lambda_2, .. \lambda_N$.

Los valores y vectores propios se clasifican respecto al orden decreciente de sus valores propios, tal que $\lambda_1 \geq \lambda_2 \geq .. \geq \lambda_N$.

Por lo tanto, la mayor variabilidad de los vectores de entrada queda contenida en los primeros vectores propios. Por este motivo, estos vectores propios se denominan vectores principales.

Computing PCA

Steps to compute the PCA transformation of a data matrix X:

- **Center the data**

$$\overline{X}$$

- **Compute the covariance matrix**

$$C$$

- **Obtain the eigenvectors and eigenvalues of the covariance matrix**

$$U \ , \ \Lambda$$

- **Project the original data in the eigenspace**

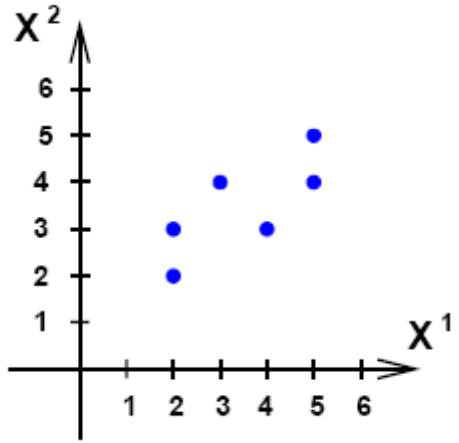
$$P = U^T \cdot \overline{X}$$

Matlab code:

```
%number of examples
N=size(X,2);
%dimension of each example
M=size(X,1);
%mean
meanX=mean(X,2);
%centering the data
Xm=X-meanX*ones(1,N);
%covariance matrix
C=(Xm*Xm')/N;
%computing the eigenspace:
[U D]=eig(C);
%projecting the centered data
   over the eigenspace
P=U'*Xm;
```

Computing PCA

Ejemplo – Matlab
X= (2,2) (2,3) (3,4) (4,3) (5,4) (5,5)



La transformación preserva la varianza global

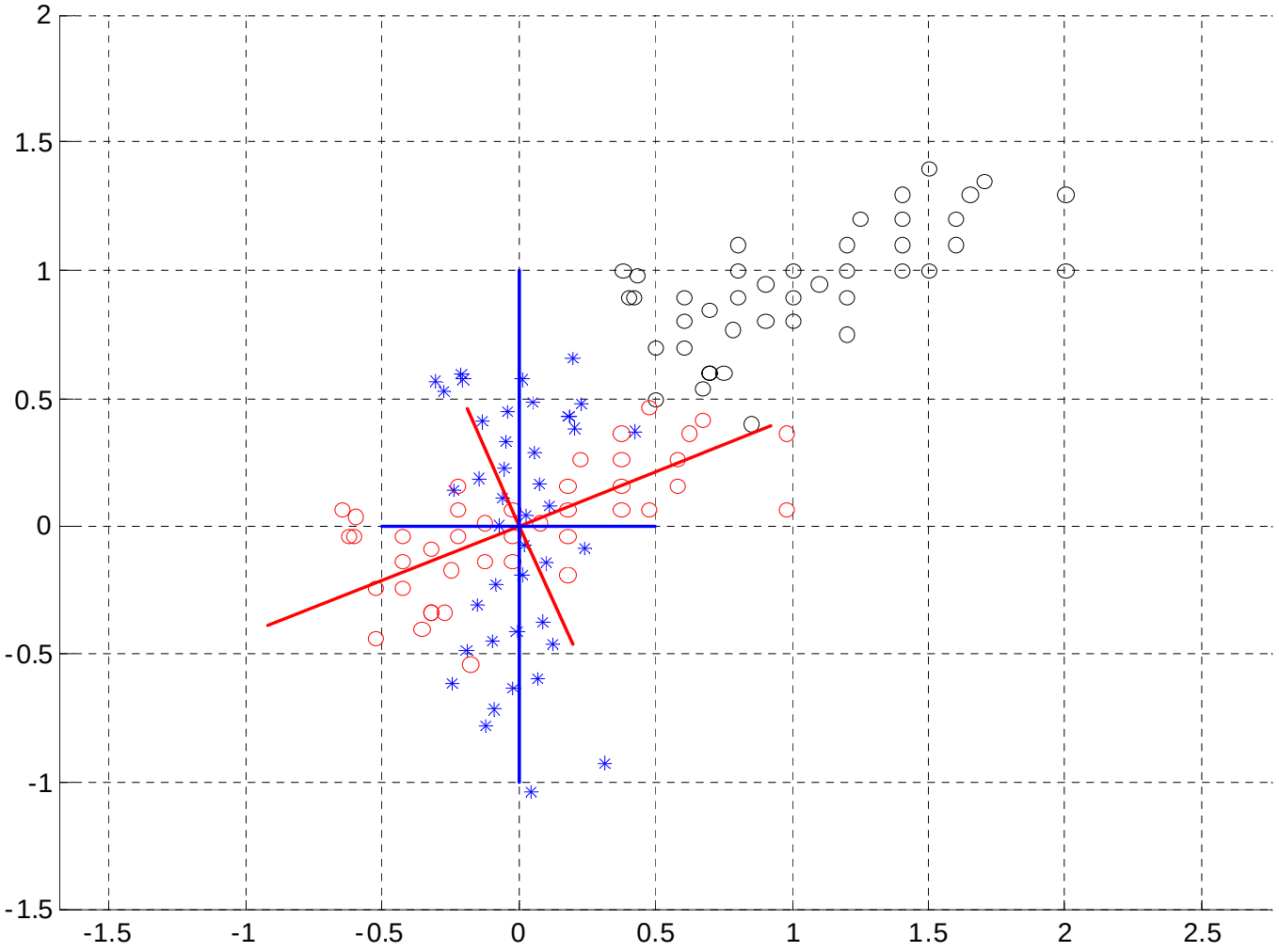
Computing PCA

Ejemplo – Matlab
Matriz de Covarianza C=(4.5 1.5; 1.5 0.5)

En este caso un autovalor es nulo
La varianza según ese vector propio es nula, lo que significa que esa nueva característica es innecesaria. La dimensionalidad de los datos en la nueva proyección es 1

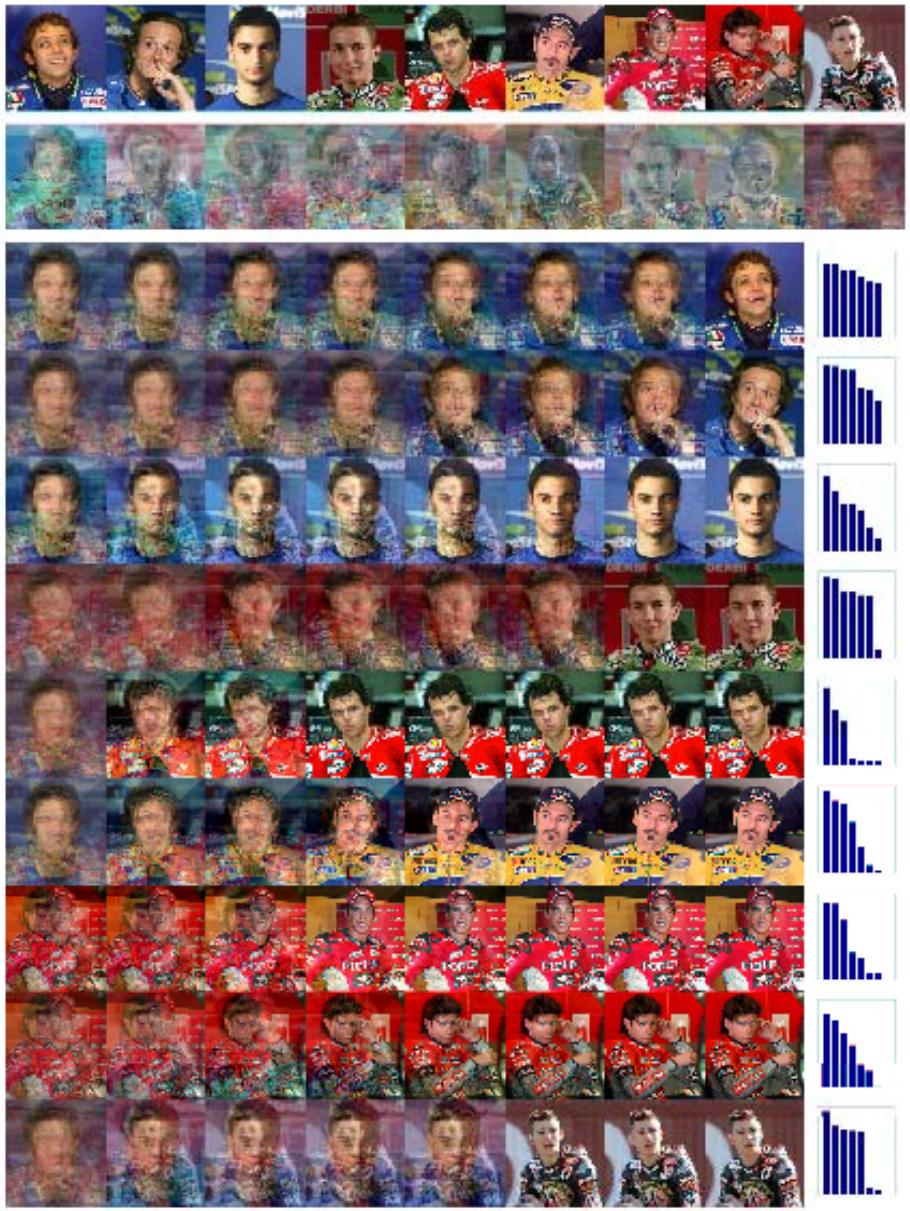
3. Selección y Extracción de características

PCA



3. Selección y Extracción de características

PCA con imágenes



Cálculo de PCA conjunto de imágenes

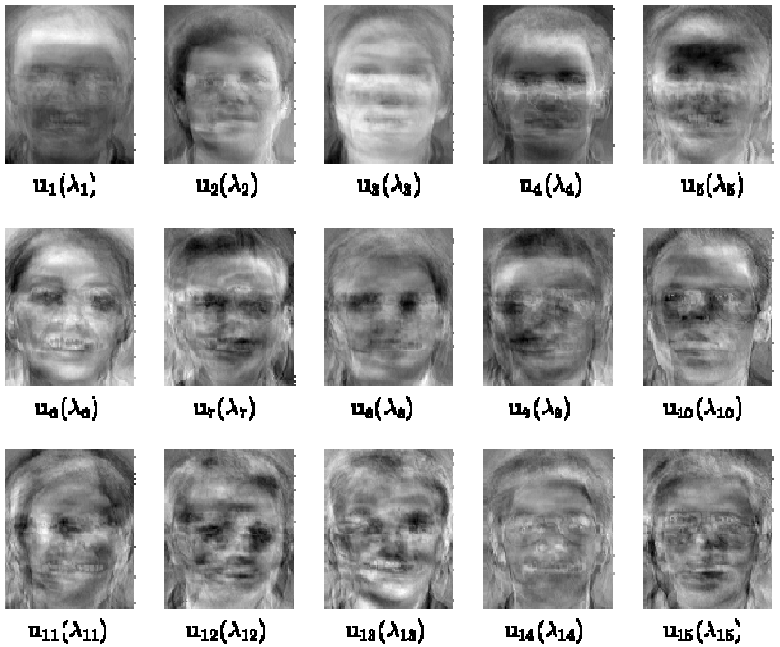
El calculo de PCA es muy utilizado al analizar imágenes. Sin embargo, si el tamaño del vector de datos o patrón M es muy grande, lo que ocurre a menudo en el caso de tratar con imágenes, la matriz de covarianzas **C** se hace muy grande y la descomposición en valores propios de **C** resulta impracticable

Pero si el numero de vectores de entrada es menor que el tamaño de estos vectores (N<M), el análisis de PCA se puede acelerar usando un método propuesto por Murakami and Kumar que se conoce como el truco de Turk and Pentland

Cuando se trabaja con Matlab se utiliza la función **pc_evectors.m** para aplicar este cálculo al conjunto de imágenes.

Reconocimiento de caras usando PCA

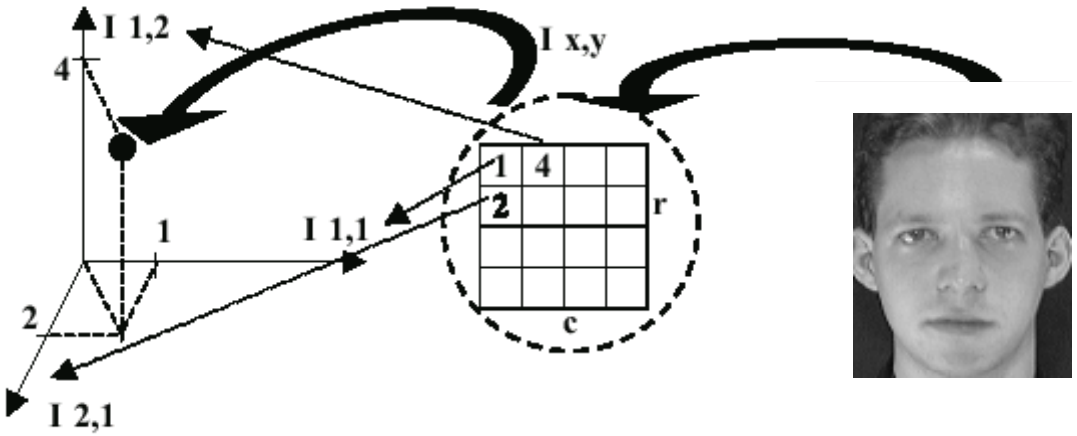
Eigenfaces for Recognition, *Turk, M. & Pentland, A. , Journal of Cognitive Neuroscience, 3, 71-86, 1991.*



FACE RECOGNITION

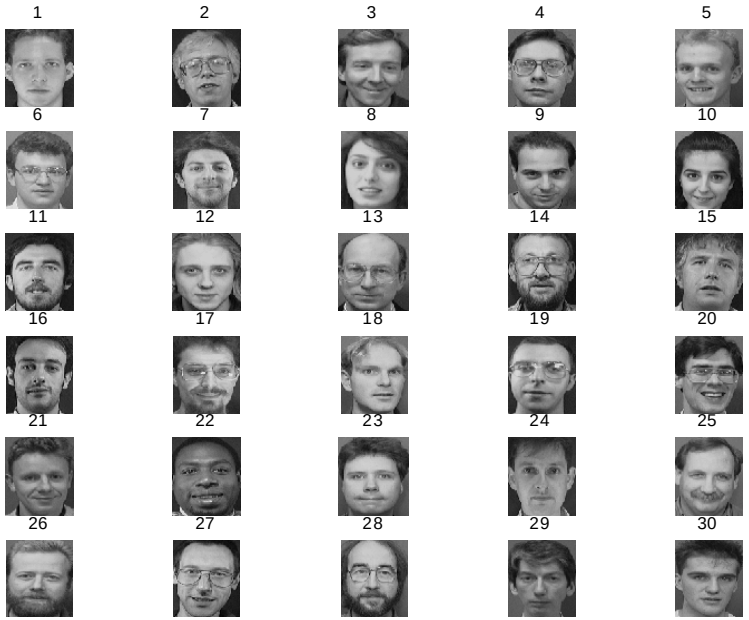
FACE DETECTION

RECONOCIMIENTO DE CARAS BASADO EN APARIENCIA



Los métodos basados en apariencia se basan en el concepto de espacio de imagen. Una imagen bidimensional **I (x,y)** puede interpretarse como un vector o punto en un espacio de elevada dimensionalidad, a menudo llamado **espacio de imagen** donde cada coordenada del espacio se corresponde con un valor del pixel en la imagen original. En general, una imagen en escala de gris con **r** filas y **c** columnas constituye un vector **x** en un espacio **m-dimensional**, siendo **m = r x c** . Con esta representación, la imagen se constituye como un vector de elevada dimension, pudiendo aplicar como clasificador un algoritmo simple como el vecino más cercano.

RECONOCIMIENTO DE CARAS BASADO EN APARIENCIA



30 members

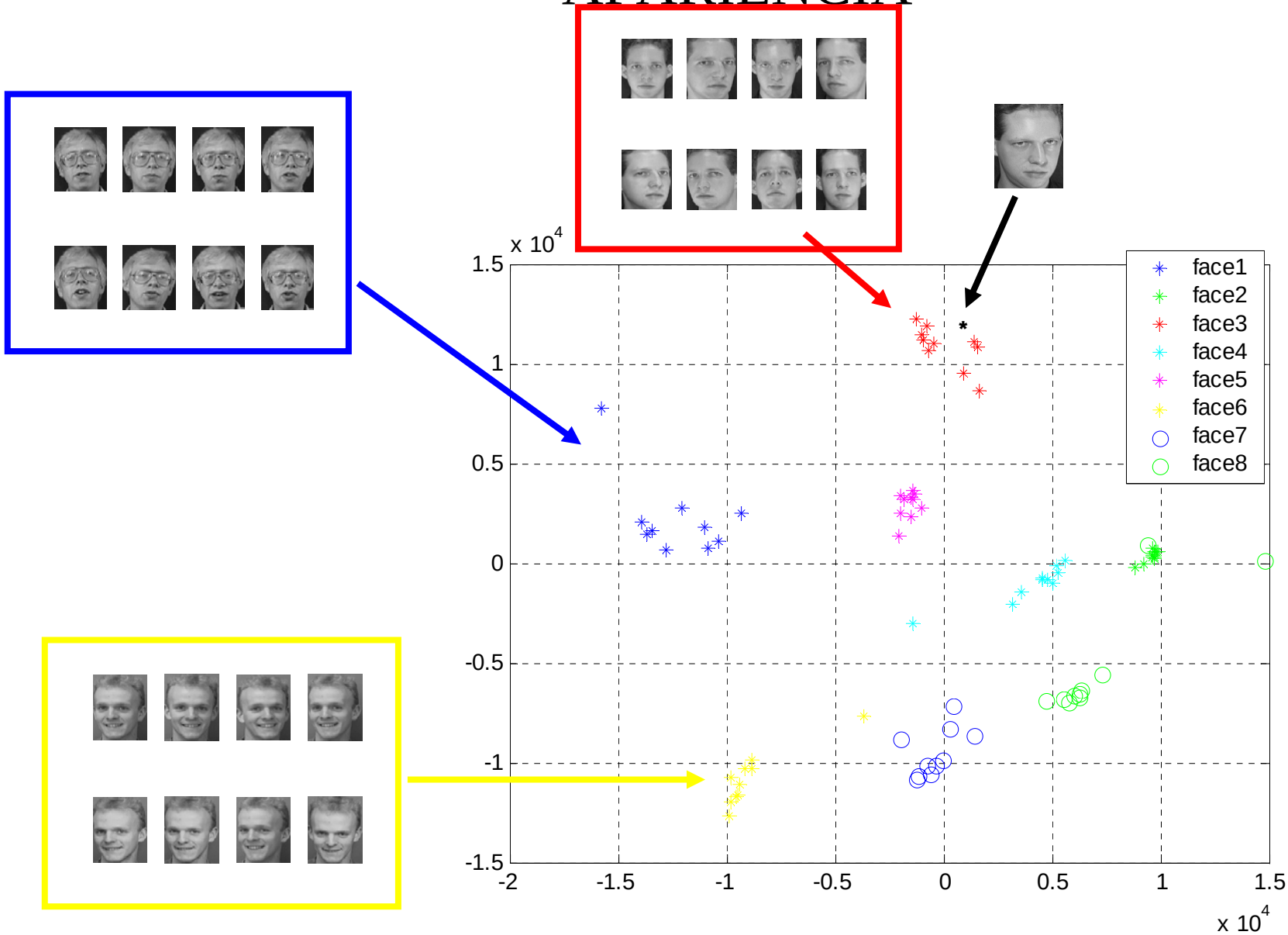
Number of pixels

$$\begin{pmatrix} 12 & 255 & 100 & \dots & 12 \\ 13 & 234 & 34 & \dots & 34 \\ 15 & 34 & 1 & \dots & 23 \\ 23 & 34 & 2 & \dots & 56 \\ 24 & 33 & 123 & \dots & 46 \\ 22 & 5 & 21 & \dots & 76 \\ 23 & 34 & 213 & \dots & 12 \\ 11 & 23 & 23 & \dots & 23 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 12 & 0 & 0 & \dots & 255 \end{pmatrix}$$

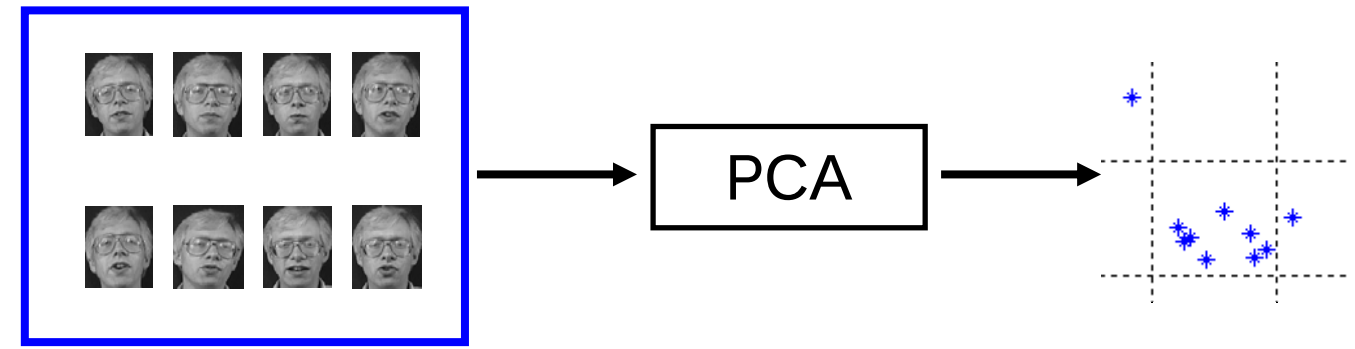
RECONOCIMIENTO DE CARAS BASADO EN APARIENCIA

Los sistemas de reconocimiento de caras no pueden implementarse directamente sobre estos espacios de tan elevada dimensión ya que haría el clasificador extraordinariamente lento.

RECONOCIMIENTO DE CARAS BASADO EN APARIENCIA



REDUCCIÓN DE CARACTERÍSTICAS



Al reducir la dimensionalidad de los datos se aceleran los cálculos sin perder mucha información.

Una técnica para reducir la dimensión de los datos es PCA

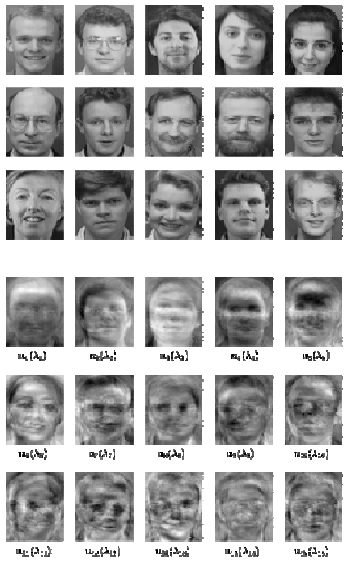
FACE RECOGNITION USING PCA

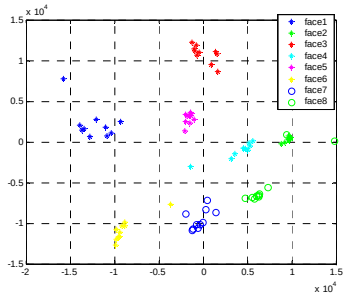
TRAINING

1. **Adquirir un conjunto inicial de caras (conjunto de entrenamiento)**

2. **Calcular la transformada PCA del conjunto de entrenamiento (vectores y valores propios) ‘eigenspace’**

3. **Proyectar las imágenes de entrenamiento sobre ese espacio de vectores propios “eigenspace”**

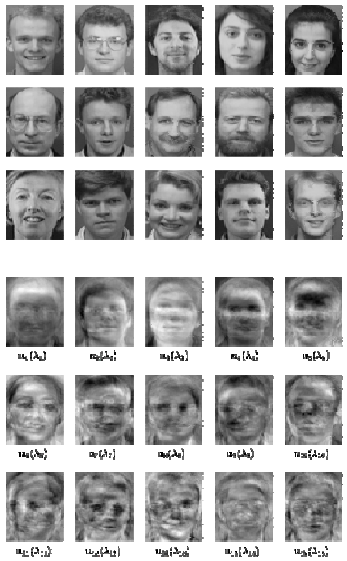


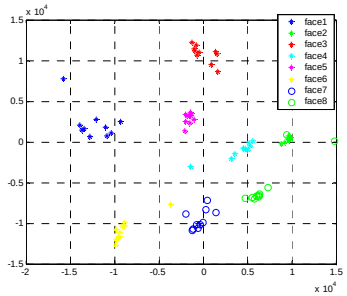


FACE RECOGNITION USING PCA and Matlab

TRAINING (matlab code)

```
%data matrix
%each column is a face
X=[x1 x2 x3 x4 x5 x6 x7 x8];
% PCA
[V,D,Average] = pc_evectors(X,Nv);
%projecting the data
P=V'*Xm;
```





FACE RECOGNITION USING PCA and Matlab (II)

TEST (TO RECOGNIZE NEW FACES)

1. Project the new image onto the "eigenspace"

2. Use a classifier to see which class is nearest to the new face

classifier.m