UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola d'Enginyeria de Barcelona Est

# Dimensionality Reduction with Principal Component Analysis (PCA)


Raúl Benítez
raul.benitez@upc.edu

Christian Mata
Christian.mata@upc.edu

# DIMENSIONALITY REDUCTION

# Big & High-Dimensional Data

- High-Dimensions = Lot of Features

Document classification

Features per document =
thousands of words/unigrams
millions of bigrams, contextual
information

Surveys -Netflix

480189 users x 17770 movies

| | movie 1 | movie 2 | movie 3 | movie 4 | movie 5 | movie 6 |
|---|---|---|---|---|---|---|
| Tom | 5 | ? | ? | 1 | 3 | ? |
| George | ? | ? | 3 | 1 | 2 | 5 |
| Susan | 4 | 3 | 1 | ? | 5 | 1 |
| Beth | 4 | 3 | ? | 2 | 4 | 2 |

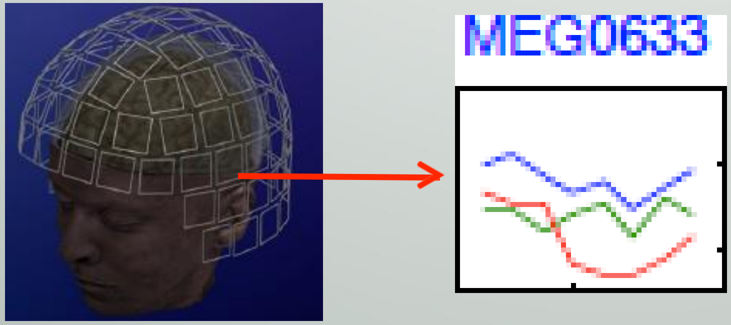
# Big & High-Dimensional Data

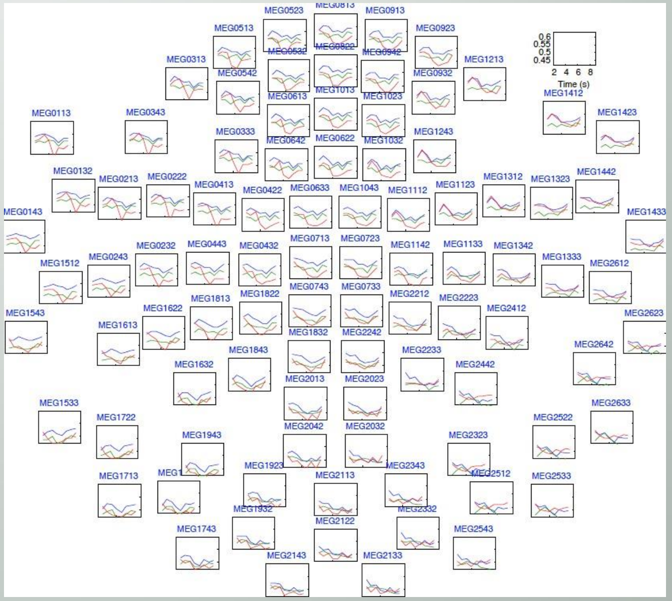
- High–Dimensions = Lot of Features

MEG Brain Imaging

120 locations x 500 time points

x 20 objects



Or any high-dimensional image data

# Big & High-Dimensional Data

- Big & High-Dimensional Data.
- Useful to learn lower dimensional representations of the data.

# Learning Representations

PCA, Kernel PCA, ICA: Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

**Useful for:**

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)
- Further processing by machine learning algorithms

# Principal Component Analysis (PCA)

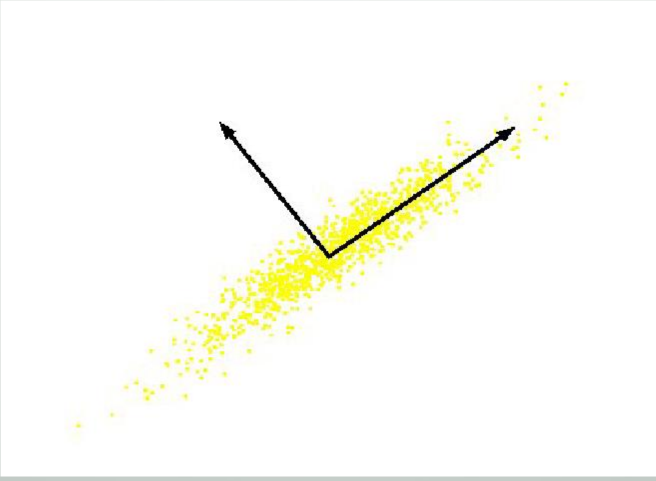
# Principal Component Analysis (PCA)

- PART 1: The idea

  What is PCA?

# Principal Component Analysis (PCA)

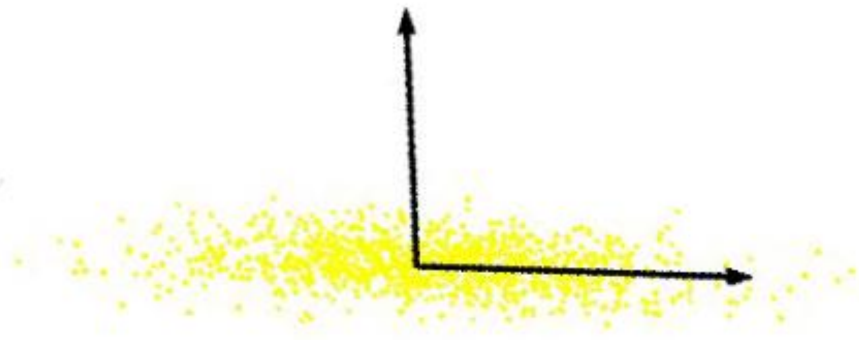
**What is PCA:** Unsupervised technique for extracting variance structure from high dimensional datasets.



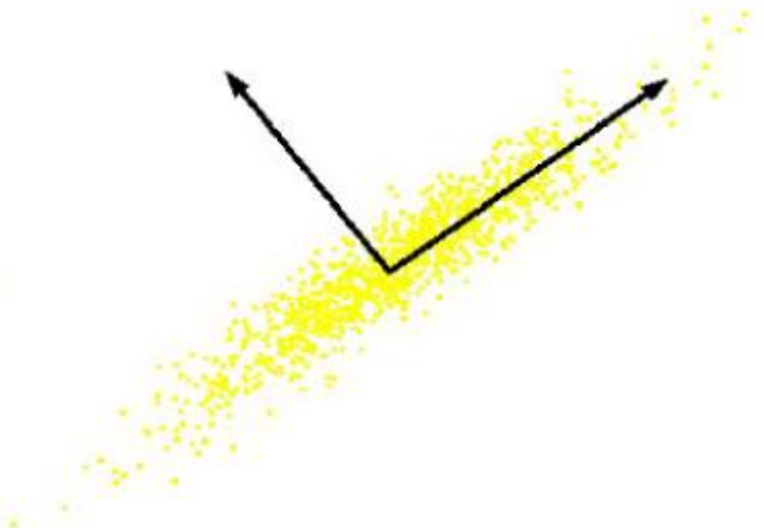
- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

# Principal Component Analysis (PCA)

Intrinsically lower dimensional than the dimension of the ambient space.



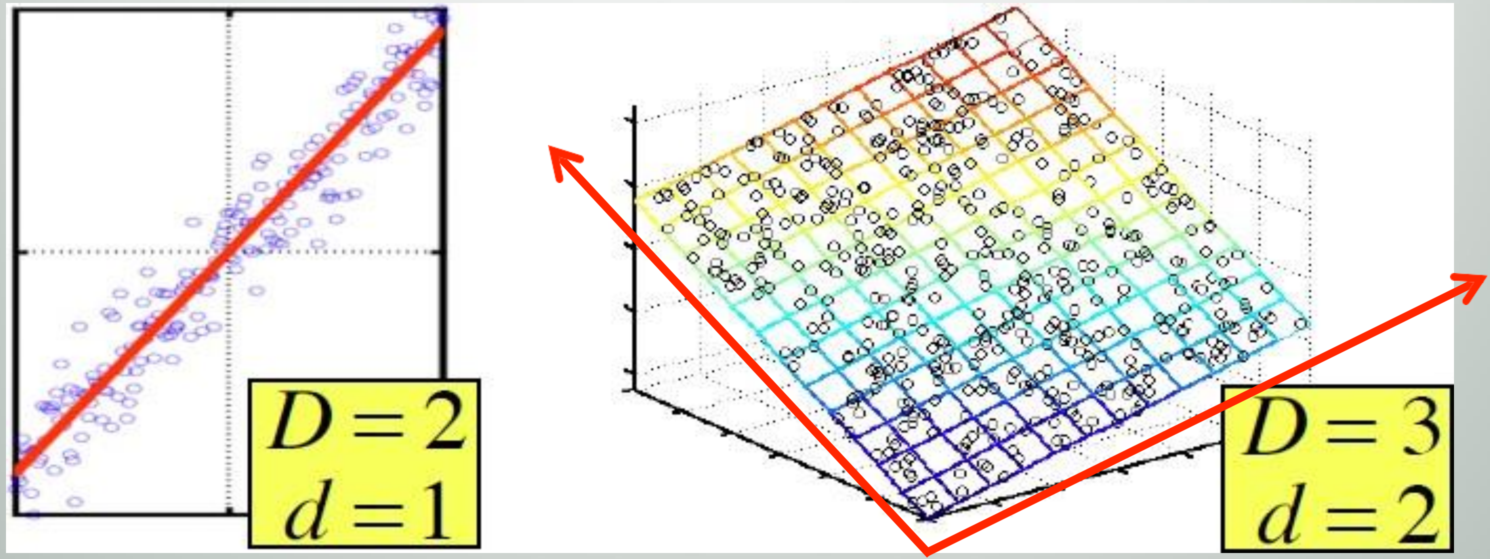
If we rotate data, again only one coordinate is more important.



**Question:**

Can we transform the features so that we only need to preserve one latent feature?

# Principal Component Analysis (PCA)



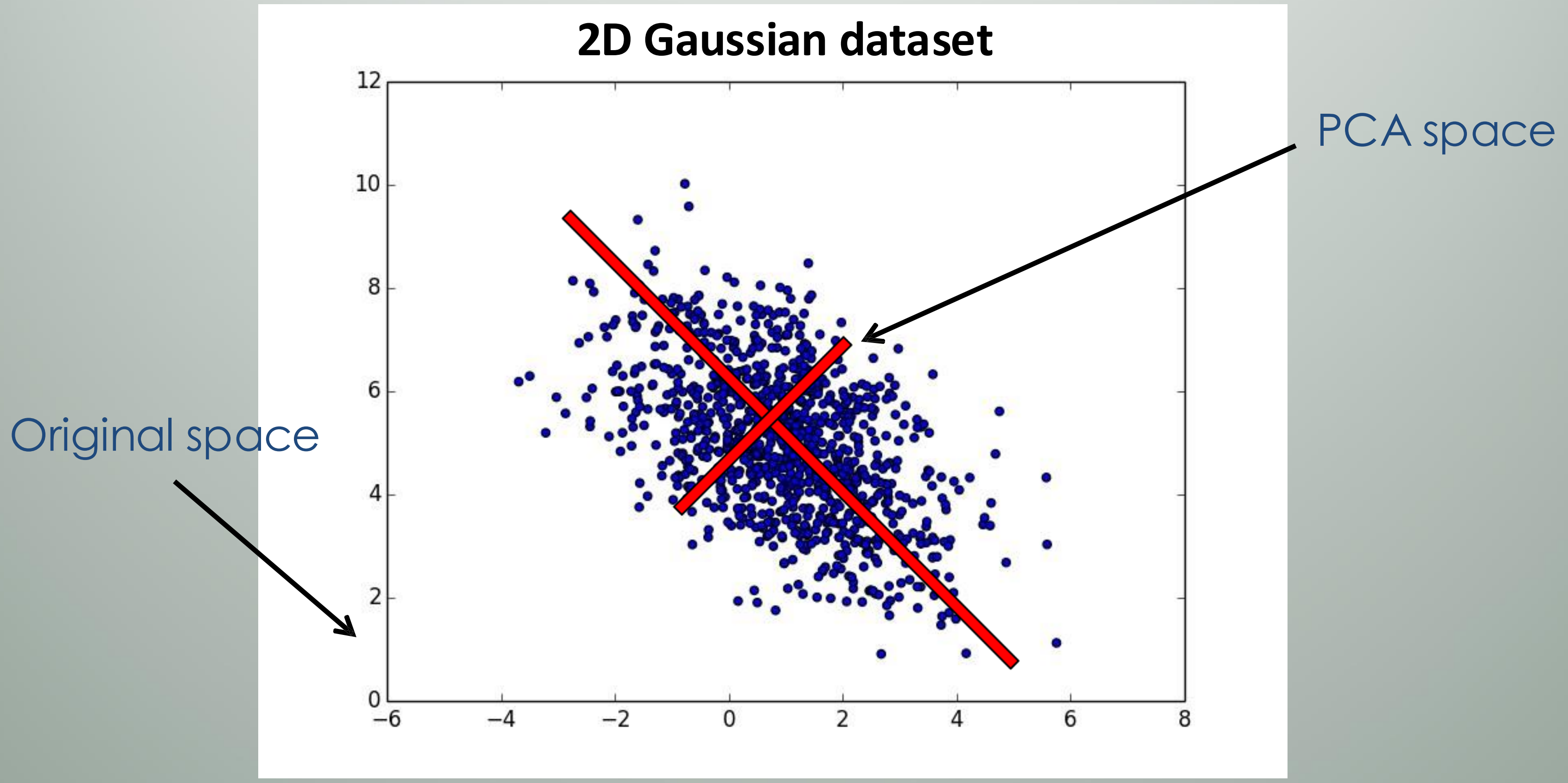

In case where data lies on or near a low d–dimensional linear subspace, axes of this subspace are an effective representation of the data.

Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

# Principal Component Analysis (PCA)

**Dimensionality reduction:**

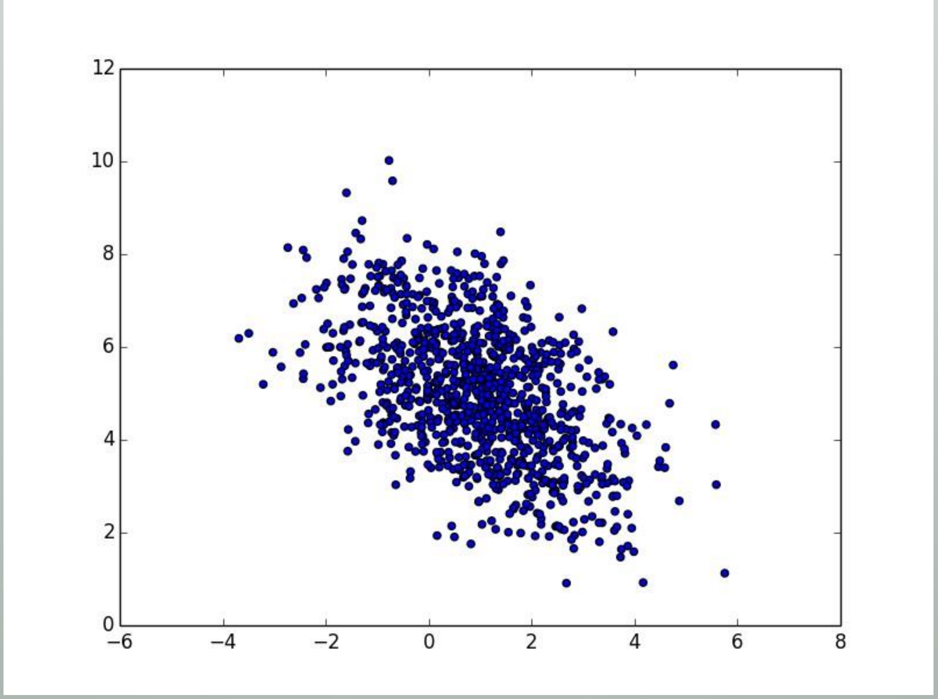
- Simplify the description of your data
- Linear transformation of the original variables



# Principal Component Analysis (PCA)

- Keep only the most relevant components (How many?)

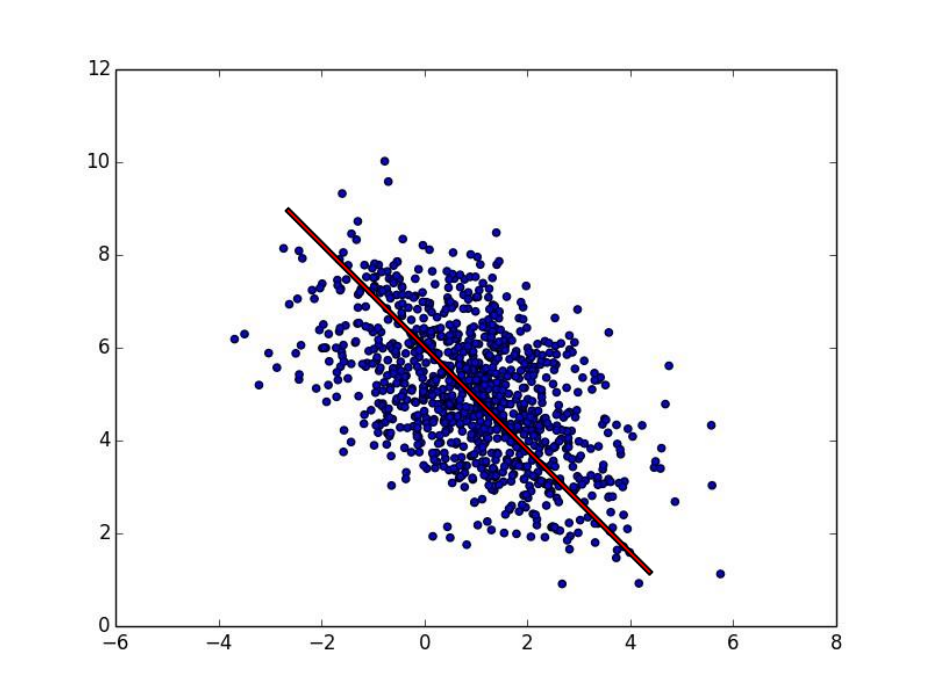
**2D Gaussian dataset**



# Principal Component Analysis (PCA)

- Keep only the most relevant components (How many?)
- Project each data point into the PCA subspace

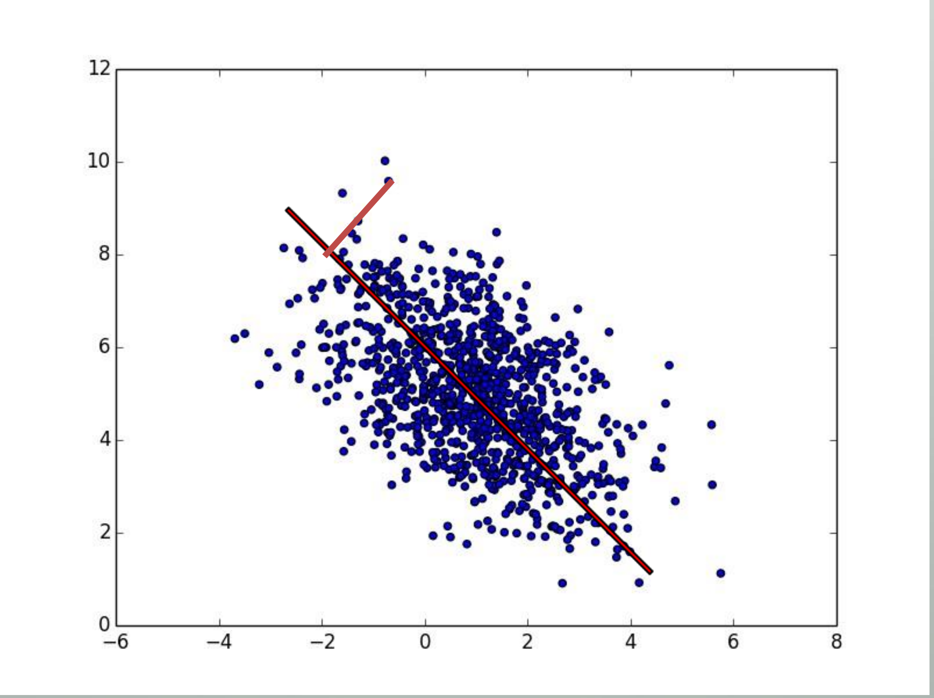
**2D Gaussian dataset**



# Principal Component Analysis (PCA)

- Keep only the most relevant components (How many?)
- Project each data point into the PCA subspace
- Use the projection as a new variable to describe your data

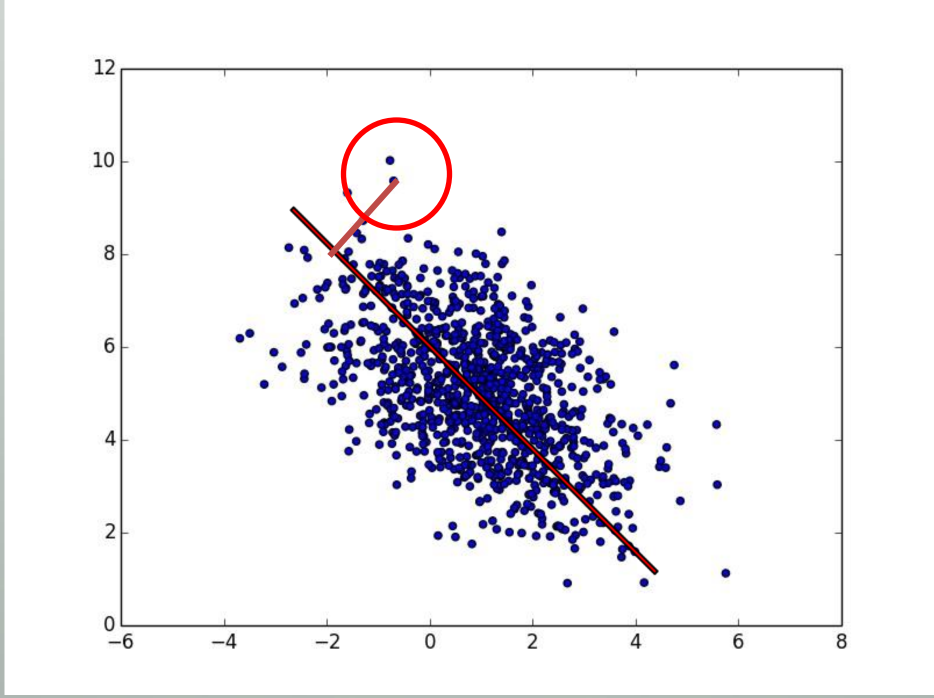
**2D Gaussian dataset**



# Principal Component Analysis (PCA)

- Keep only the most relevant components (How many?)
- Project each data point into the PCA subspace
- Use the projection as a new variable to describe your data

**2D Gaussian dataset**



# Principal Component Analysis (PCA)

- PART 1: The idea

  What is PCA?

- PART 2: The math

  How is it formulated?

# Mathematical formulation:

**Data:** n variables, m observations

$$
\begin{array}{ccc}
 & \text{variables} & \\
A_{m\times n} = & \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_n^1 \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ x_1^3 & x_2^3 & \cdots & x_n^3 \\ \cdots & & & \\ x_1^m & x_2^m & \cdots & x_n^m \end{pmatrix} & \text{observations}
\end{array}
$$

**Sample covariance matrix of data:**

$$C_{n\times n} = (A - \bar{A})^T (A - \bar{A})$$

# Procedure

**1. Diagonalize covariance matrix:**

$$C \cdot \vec{v}_i = \lambda_i \cdot \vec{v}_i, \; i = 1 \ldots n$$

Eigenvectors $\vec{v}_i$ : Coordinates of the PCA space

Eigenvalues $>_i$ : Relevance of each PCA coordinate

# Procedure

**1. Diagonalize covariance matrix:**

$$C \cdot \vec{v}_i = \lambda_i \cdot \vec{v}_i,\ i = 1 \ldots n$$

Eigenvectors $\vec{v}_i$ : Coordinates of the PCA space

Eigenvalues $>_i$ : Relevance of each PCA coordinate

**2. Keep PCA components by the variance of data explained by each component:**

$$100 \leftarrow \frac{>_i}{\overset{n}{\underset{1=1}{}} >_i}$$

# Procedure

**1. Diagonalize covariance matrix:**

$$C \cdot \vec{v}_i = \lambda_i \cdot \vec{v}_i, \;\; i = 1 \ldots n$$

Eigenvectors $\vec{v}_i$ : Coordinates of the PCA space

Eigenvalues $>_i$ : Relevance of each PCA coordinate

**2. Keep PCA components by the variance of data explained by each component:**

$$100 \leftarrow \frac{>_i}{{}^{n}_{1=1} >_i}$$

**3. Project data into PCA subspace:** $$P_i = A \cdot v_i$$

Projection of data into the PCA coordinate associated to eigenvector $\vec{v}_i$

# Example 1
# Face recognition

# Example 1: Face recognition

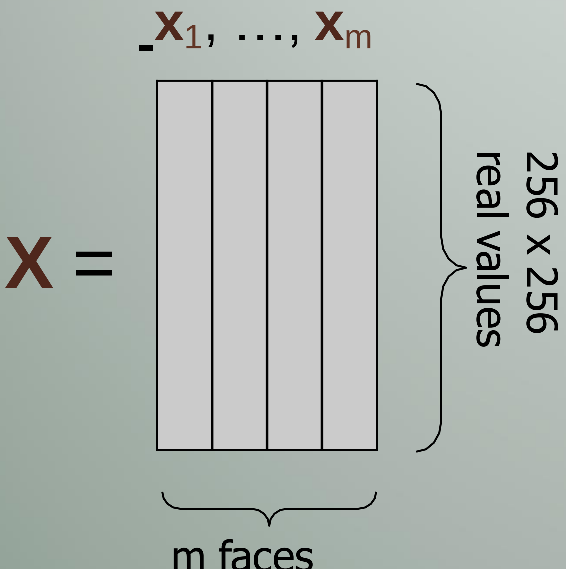
- Want to identify specific person, based on facial image
- Robust to glasses, lighting,…

®Can’t just use the given 256 x 256 pixels

# Example 1: Face recognition

**Method:** Build one PCA database for the whole dataset and then classify based on the weights.

- Example data set: Images of faces
  - Famous Eigenface approach
    [Turk & Pentland], [Sirovich & Kirby]
- Each face **x** is …
- 256 · 256 values (luminance at location)
  - **x** in $^{256 \cdot 256}$ (view as 64K dim vector)




- Form **X** = [ **x**$_1$ , …, **x**$_m$ ] **centered** data mtx
- Compute © = **XX**$^T$
- Problem: © is 64K · 64K … HUGE!!!

# Example 1: Face recognition

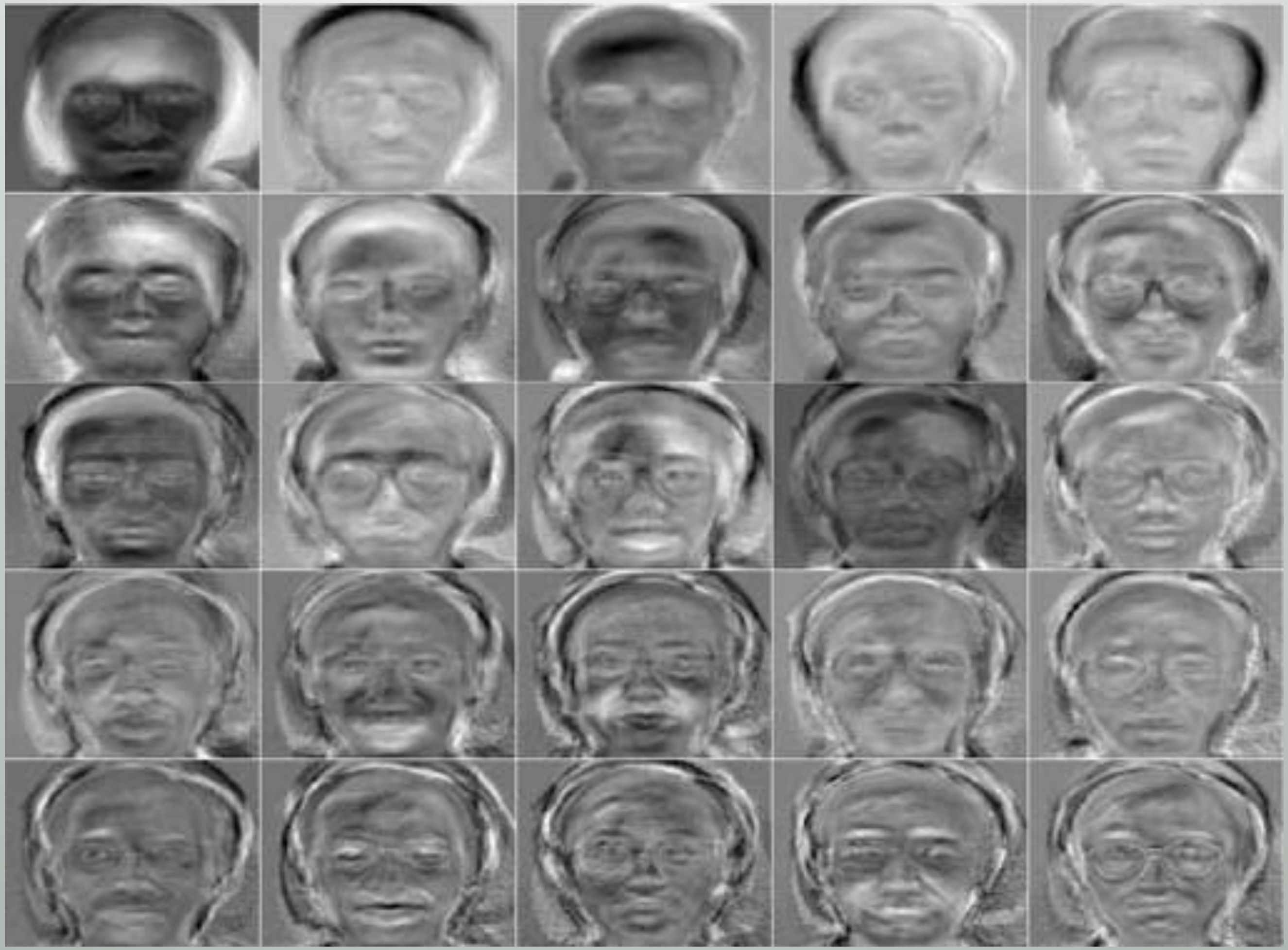
## Computational Complexity

- Suppose m instances, each of size N
  - Eigenfaces: m=500 faces, each of size N=64K

- Given $N \cdot N$ covariance matrix ©, can compute
  - all N eigenvectors/eigenvalues in $O(N^3)$
  - first k eigenvectors/eigenvalues in $O(k\ N^2)$

- But if N=64K, EXPENSIVE!

# Example 1: Face recognition



# Example 1: Face recognition

## Reconstructing

- … faster if train with…
  - only people w/out glasses
  - same lighting conditions

# Example 1: Face recognition

## Shortcomings

- Requires carefully controlled data:
  - All faces centered in frame
  - Same size
  - Some sensitivity to angle
- Alternative:
  - "Learn" one set of PCA vectors for each angle
  - Use the one with lowest error
- Method is completely knowledge free
  - (sometimes this is good!)
  - Doesn't know that faces are wrapped around 3D objects (heads)
  - Makes no effort to preserve class distinctions

# Example 2

## Facial expression recognition

# Example 2: Facial expression recognition

## Happiness subspace

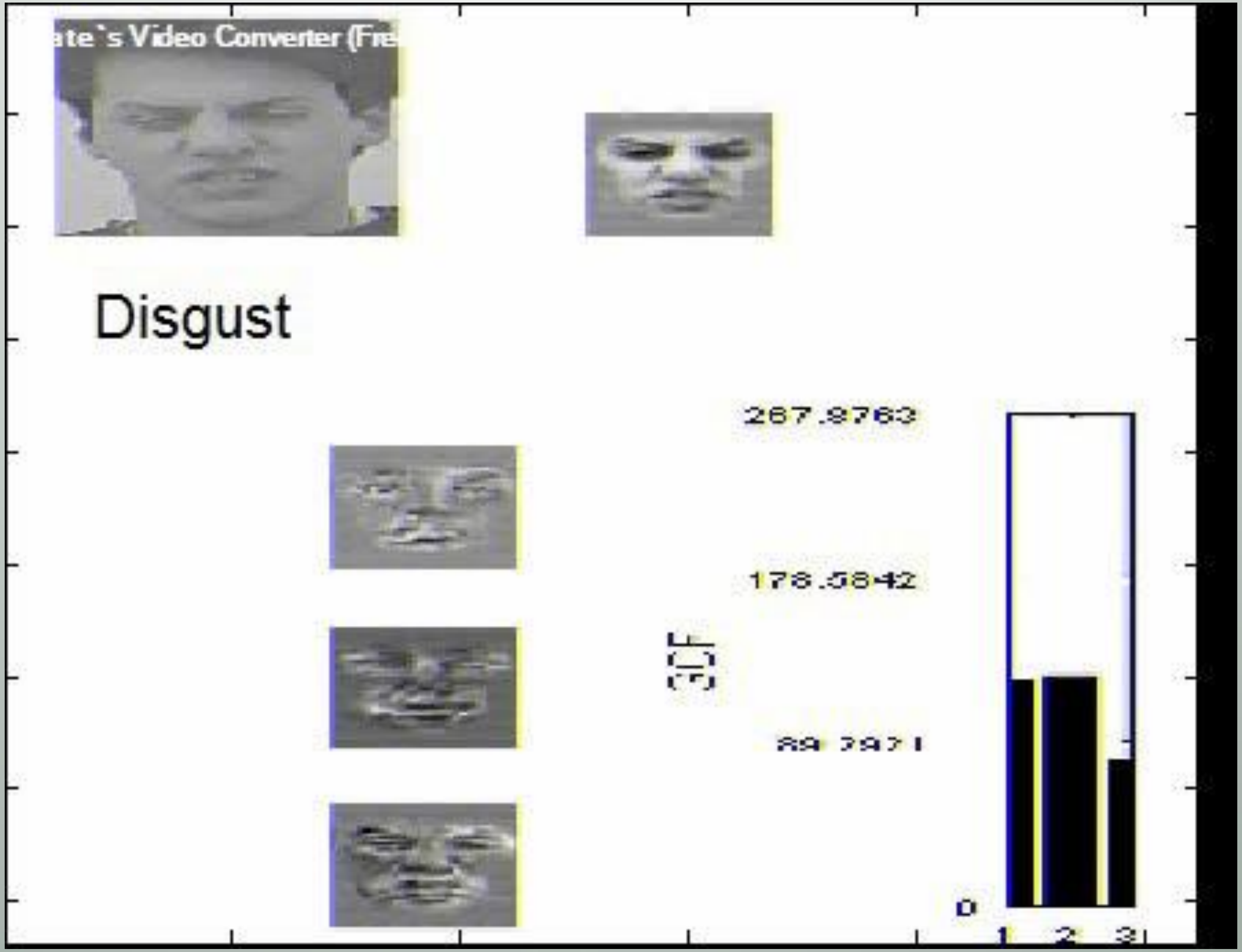
# Example 2: Facial expression recognition

## Disgust subspace

# Example 2: Facial expression recognition

# Example 2: Facial expression recognition

# Example 2: Facial expression recognition



# Example 3

## Image compression

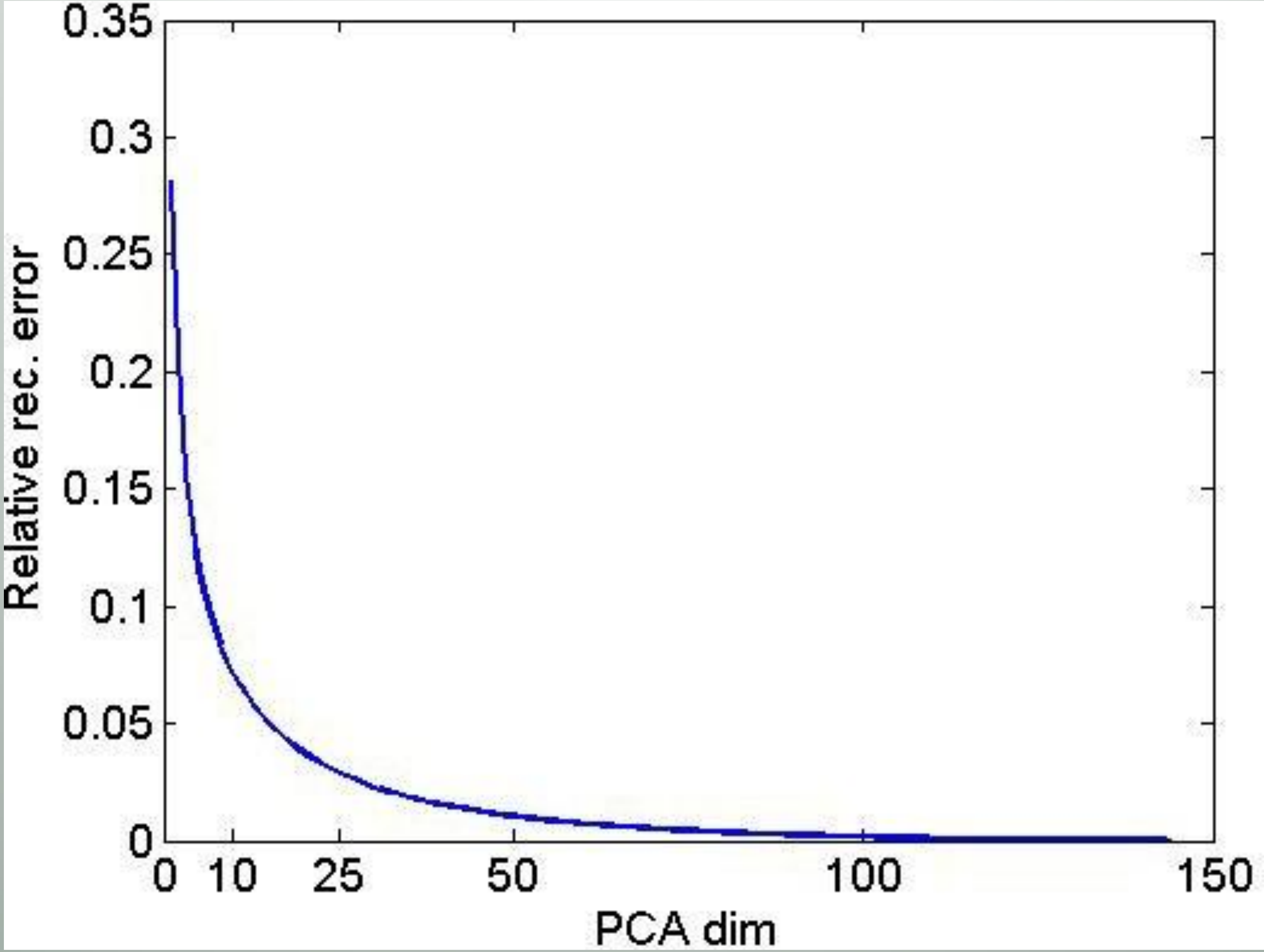
# Example 3: Image compression

**Original Image**

- Divide the original 372x492 image into patches:
  - Each patch is an instance that contains 12x12 pixels on a grid
- View each as a 144–D vector

# Example 3: Image compression

**$L_2$ error and PCA dim**



# Example 3: Image compression

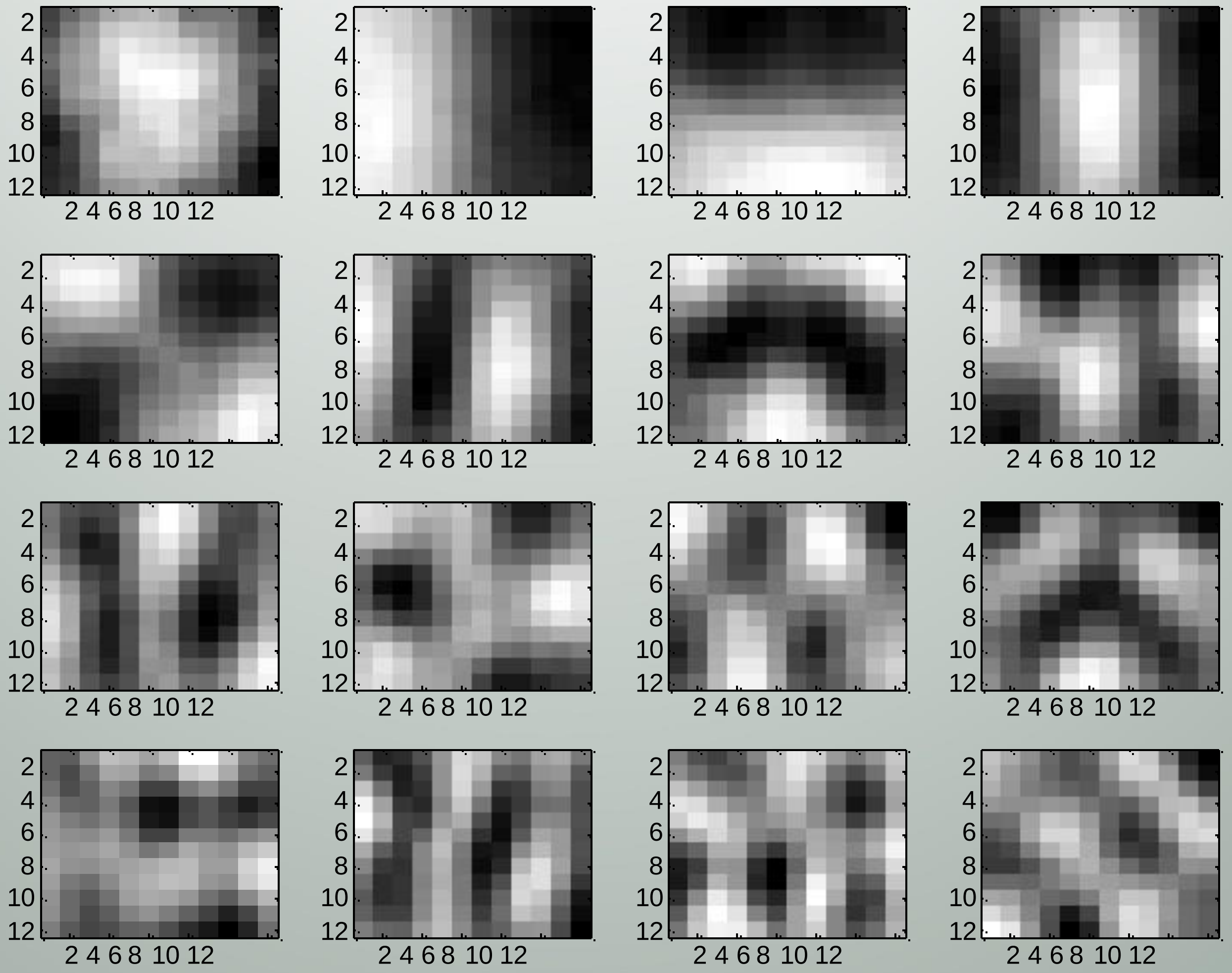
**PCA compression: 144D ) 60D**

# Example 3: Image compression

**PCA compression: 144D ) 16D**

# Example 3: Image compression

## 16 Most important eigenvectors

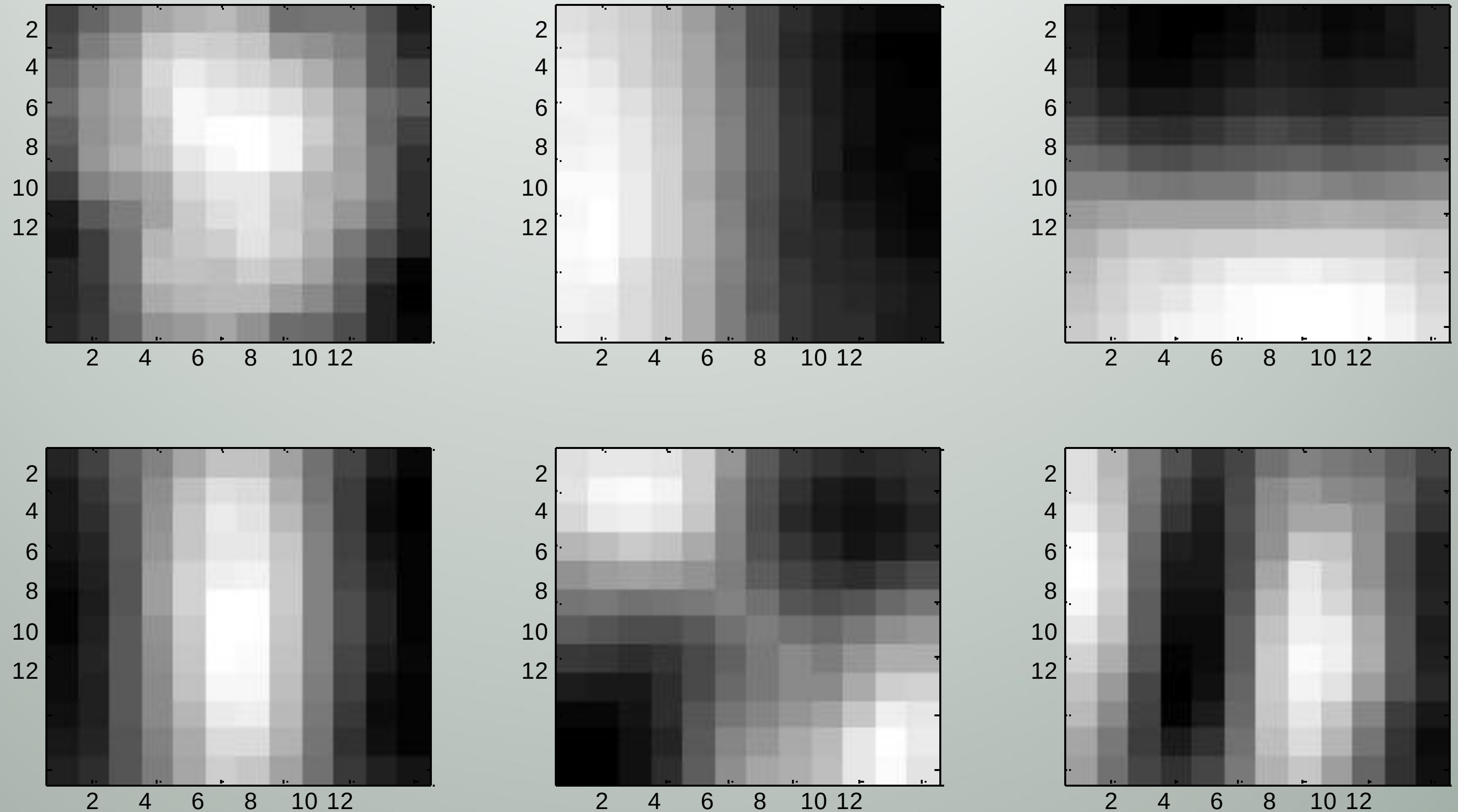


# Example 3: Image compression

**PCA compression: 144D ) 6D**

# Example 3: Image compression

## 6 Most important eigenvectors

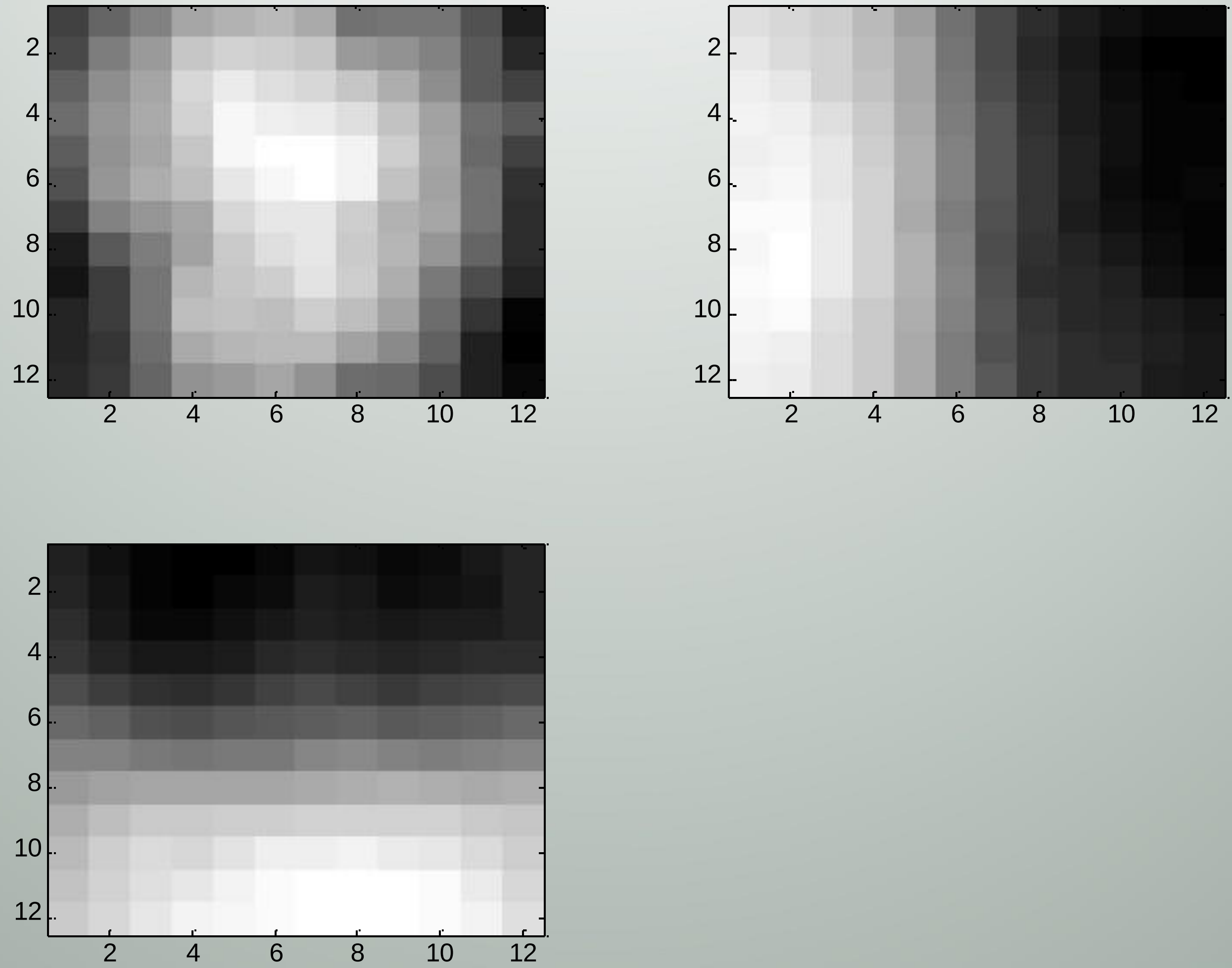


# Example 3: Image compression

**PCA compression: 144D ) 3D**

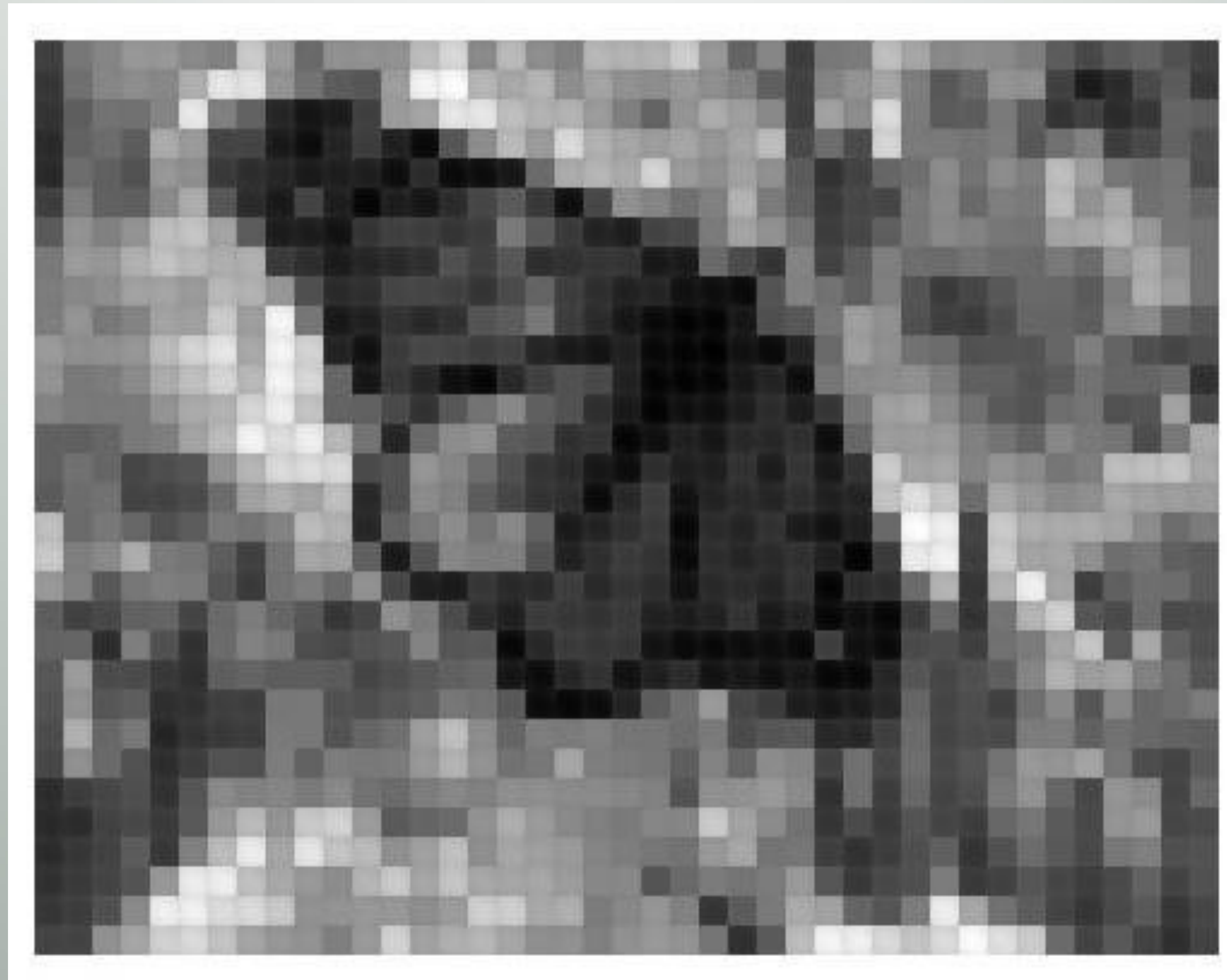
# Example 3: Image compression

## 3 Most important eigenvectors



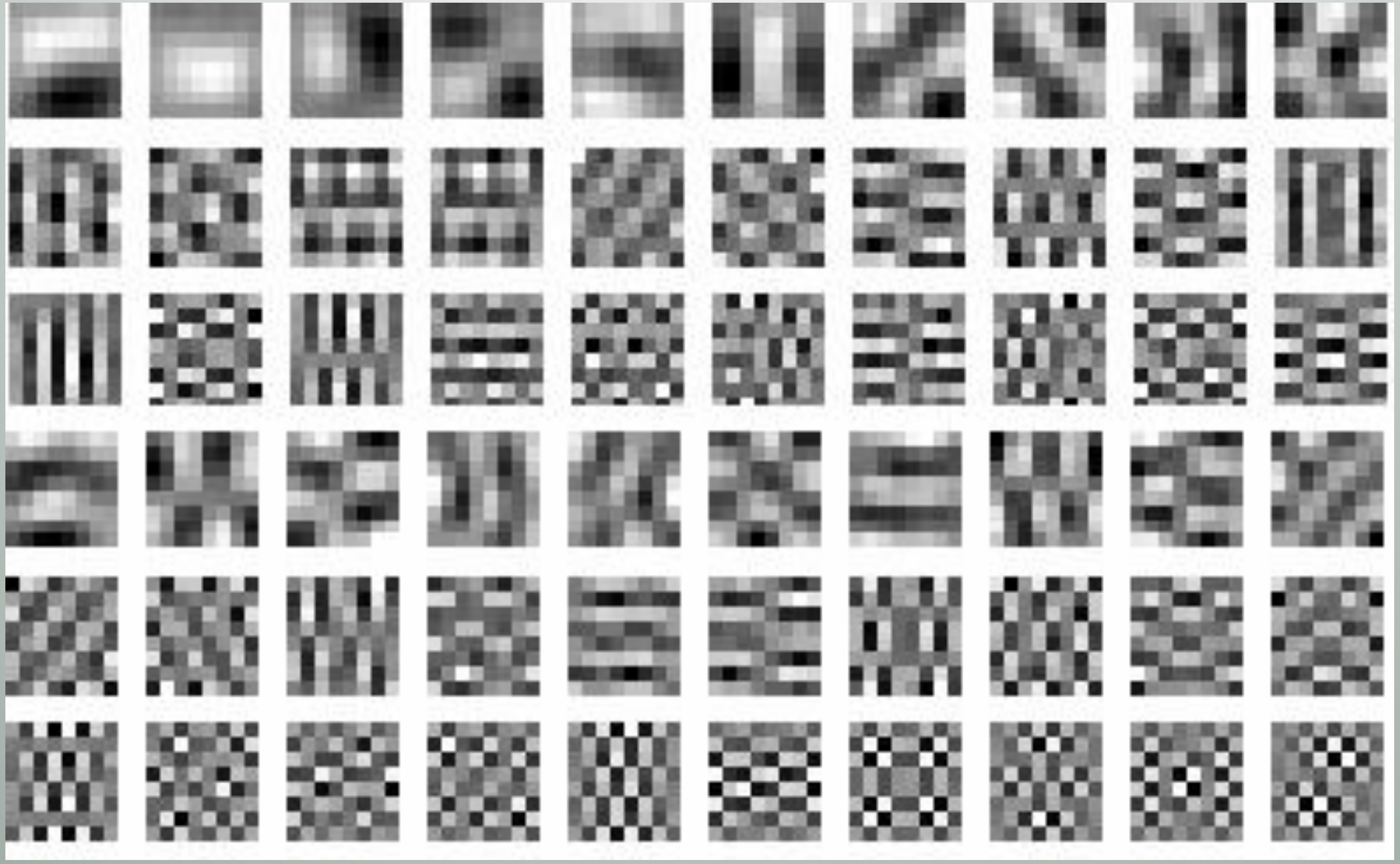
# Example 3: Image compression

**PCA compression: 144D ) 1D**



# Example 3: Image compression

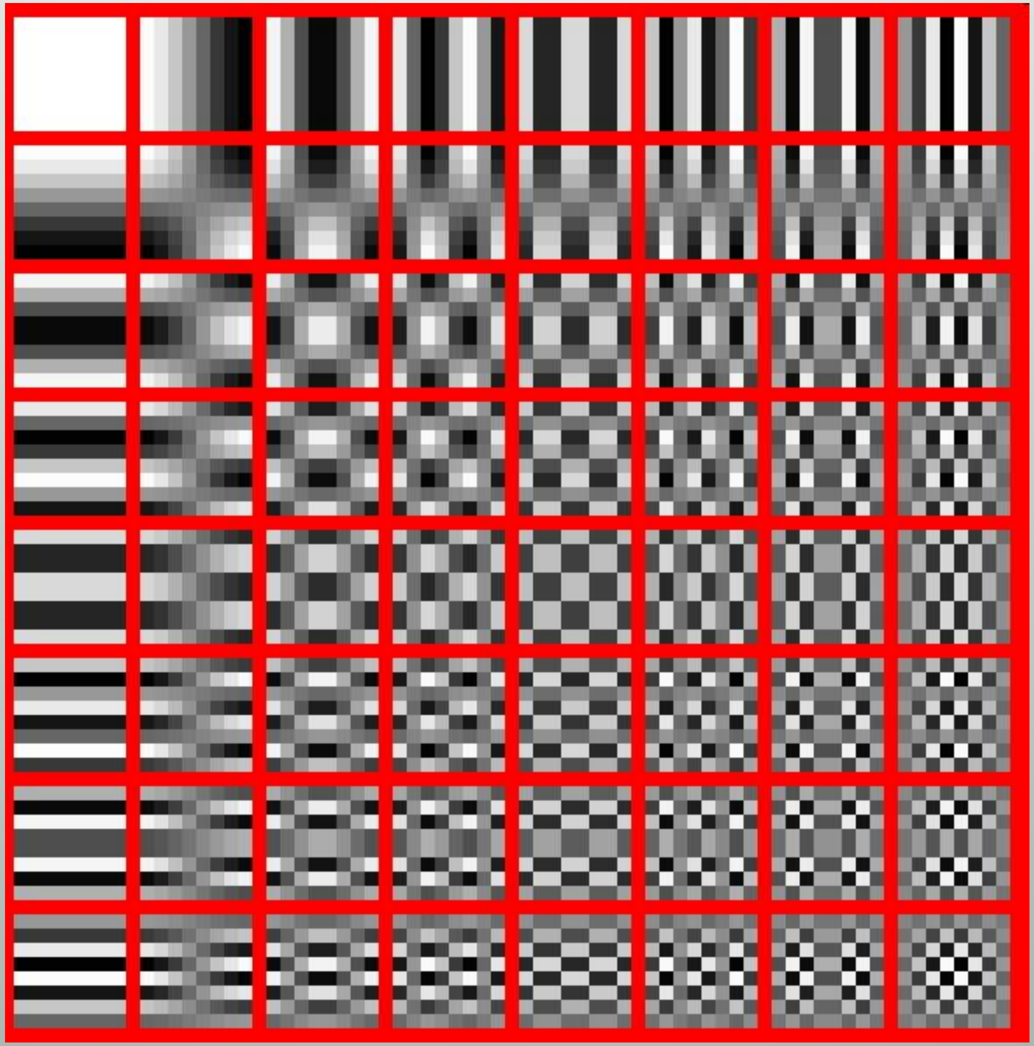
**60 Most important eigenvectors**



Looks like the discrete cosine bases of JPG!...

# Example 3: Image compression

**2D Discrete Cosine Basis**



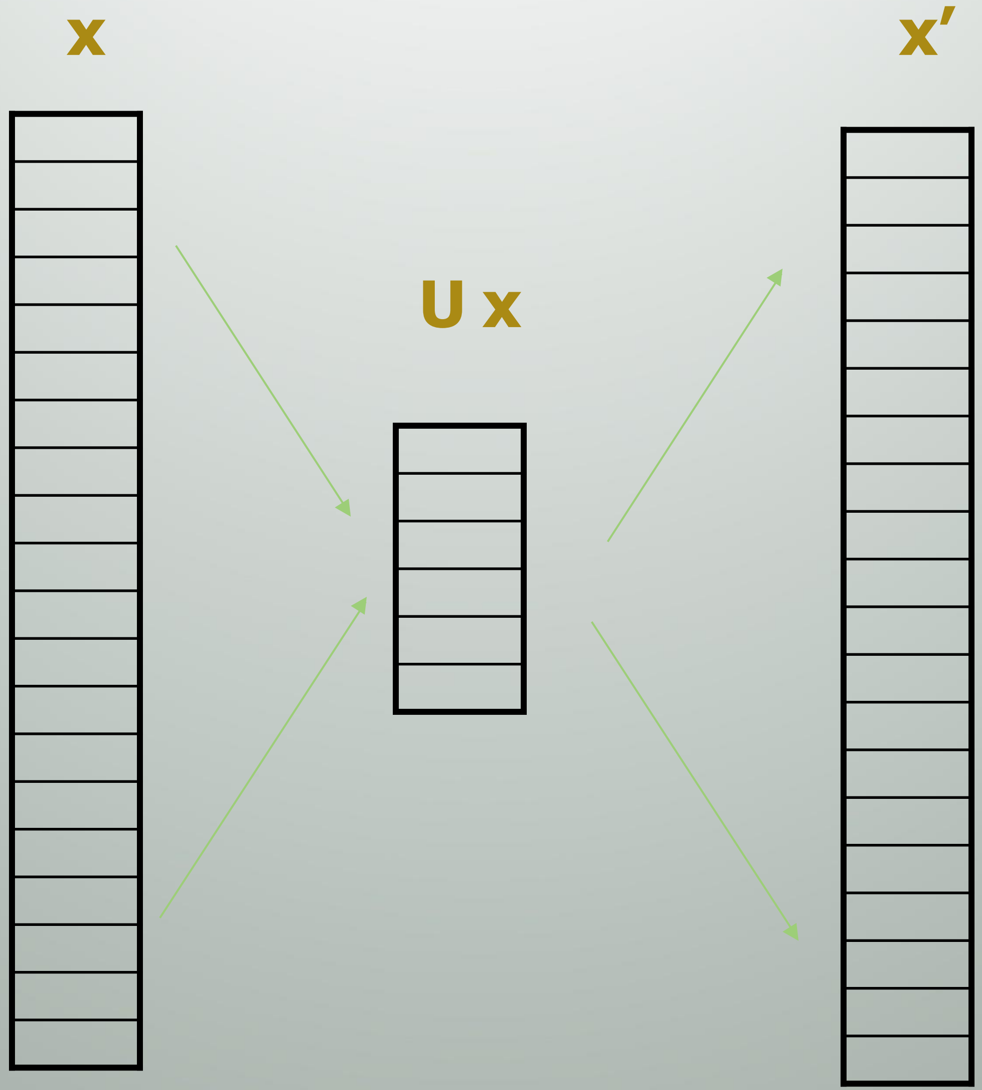
http://en.wikipedia.org/wiki/Discrete_cosine_transform

# Example 4

# Noise filtering

# Example 4: Noise filtering

**Noise Filtering, Auto-Encoder…**



# Example 4: Noise filtering

**Noisy image**

# Example 4: Noise filtering

**Denoised image using 15 PCA components**

# Principal Component Analysis (PCA)

- PART 1: The idea

  What is PCA?

- PART 2: The math

  How is it formulated?

- PART 3: The code

  How to perform PCA in Python?

# Python Implementation

## Approach 1: do it manually (scipy)

```python
# Obtain covariance matrix:
A1 = A - A.mean(0)
matcov = dot(A1.transpose(),A1)

# Diagonalization of covariance matrix:
valp,vecp = linalg.eig(matcov)

ind_creciente = argsort(valp) # sort eigenvalues
```

## Approach 2: Use sklearn tools

```python
# Use sklearn rutines to obtain PCA projection:

from sklearn.decomposition import PCA

# Projection into 1d PCA space:
pca = PCA(n_components=1)
#X_r = pca.fit(A).transform(A)
X_r = pca.fit_transform(A)
```

# More information

## Numpy linear algebra-eigenvalues:

http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eig.html

## Sklearn tools:

http://scikit- learn.org/stable/modules/generated/sklearn.decomposition.PCA.html