School of Computer Science
Carnegie Mellon

10-701 Introduction to Machine Learning

Principal Component Analysis and Dimensionality Reduction

Matt Gormley
Lecture 14
October 24, 2016

Readings:
Bishop Ch. 12
Murphy Ch. 12

Reminders

- Homework 3:
  - due 10/24/16 (tonight)

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

Slide from Nina Balcan

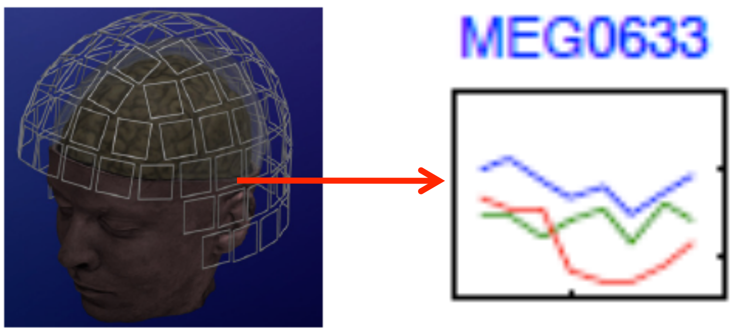
Big & High-Dimensional Data

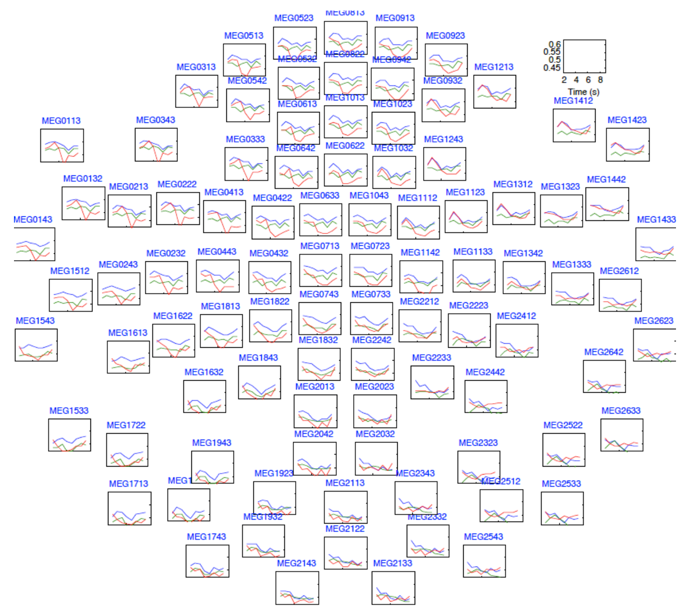
- High-Dimensions = Lot of Features

MEG Brain Imaging

120 locations x 500 time points

x 20 objects





Or any high-dimensional image data

- Big & High-Dimensional Data.

- 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.

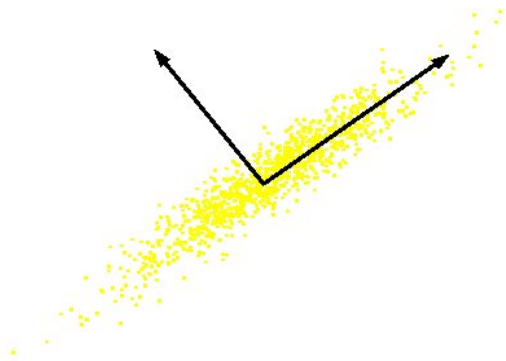
Useful for:

- 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

Slide from Nina Balcan

Principal Component Analysis (PCA)

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

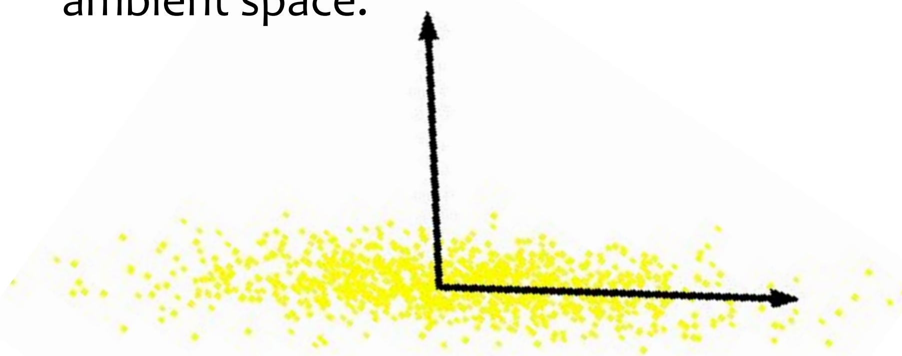


- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

Slide from Nina Balcan

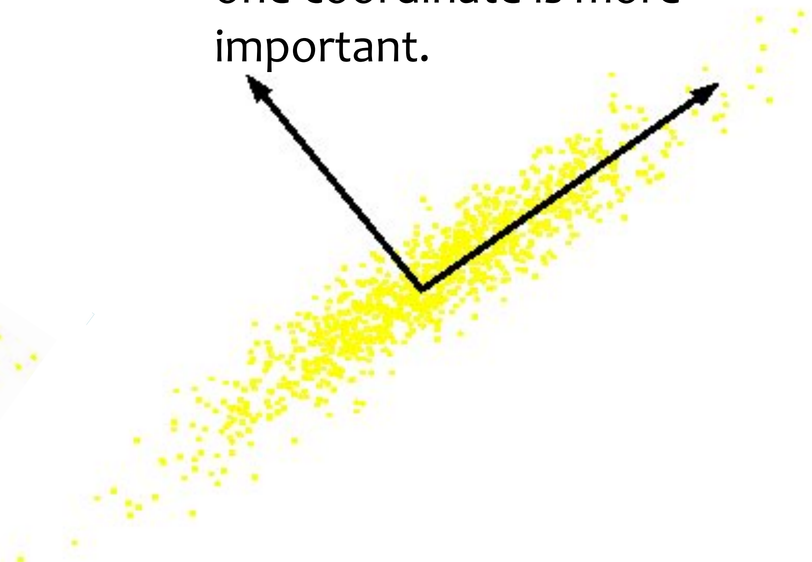
Principal Component Analysis (PCA)

Intrinsically lower dimensional than the dimension of the ambient space.



Only one relevant feature

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

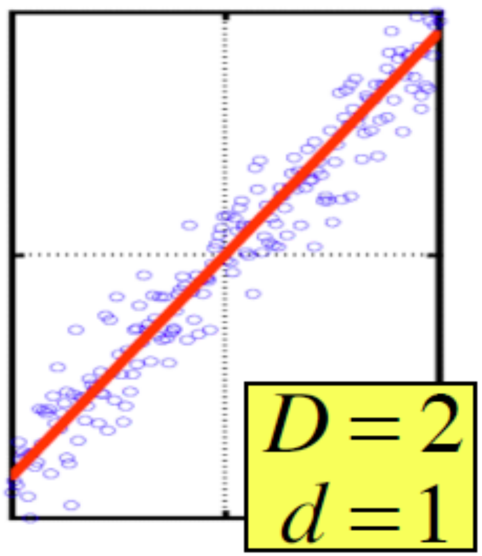


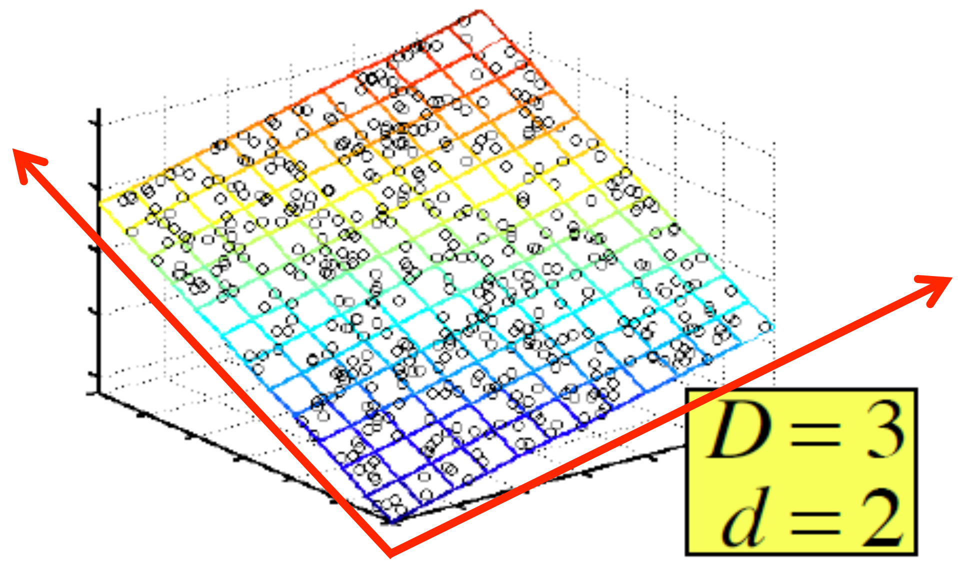
Both features are relevant

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

Slide from Nina Balcan

Principal Component Analysis (PCA)



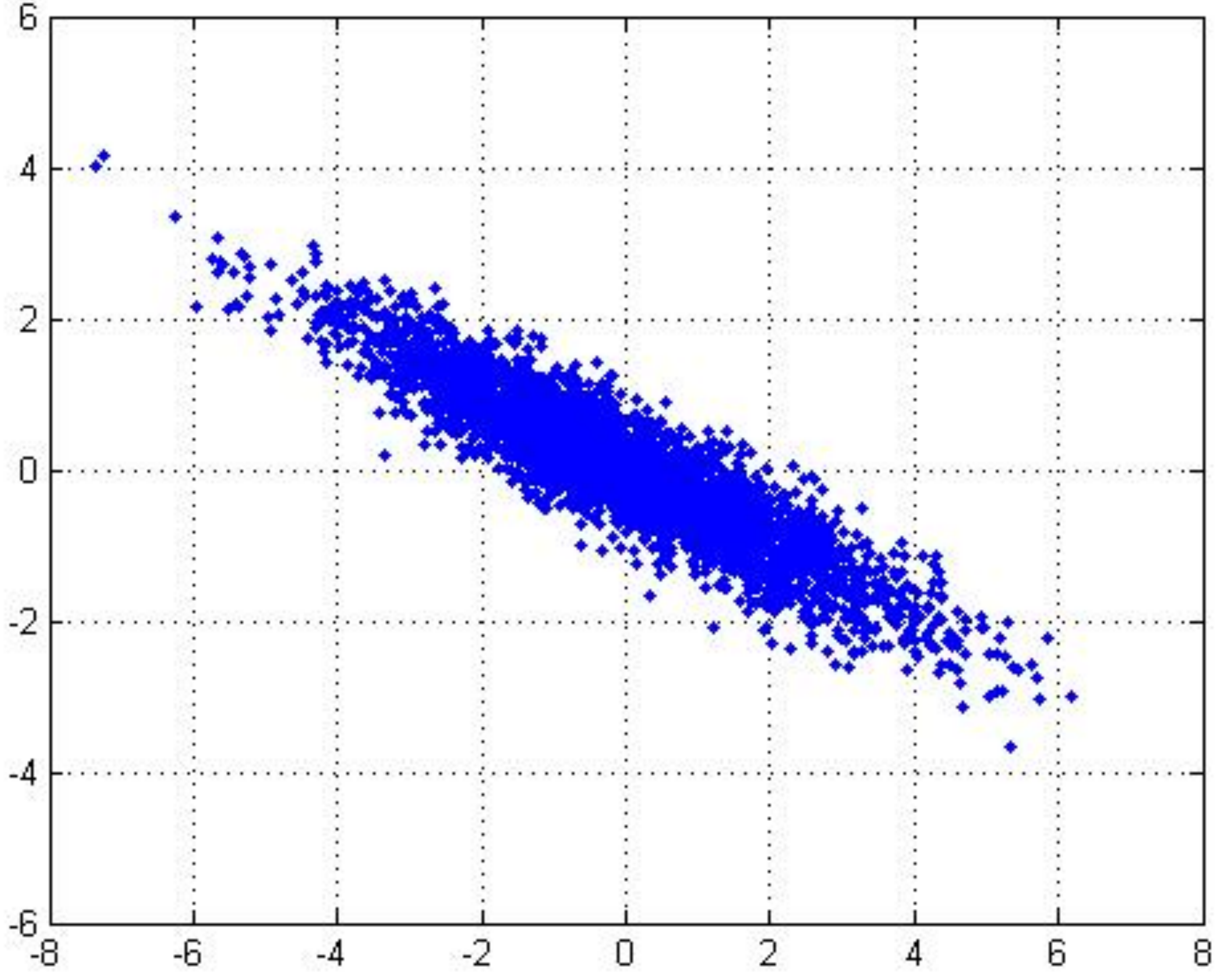


In case where data lies on or near a low d-dimensional linear subspace, axes of this subspace are an effective representation of the data.

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

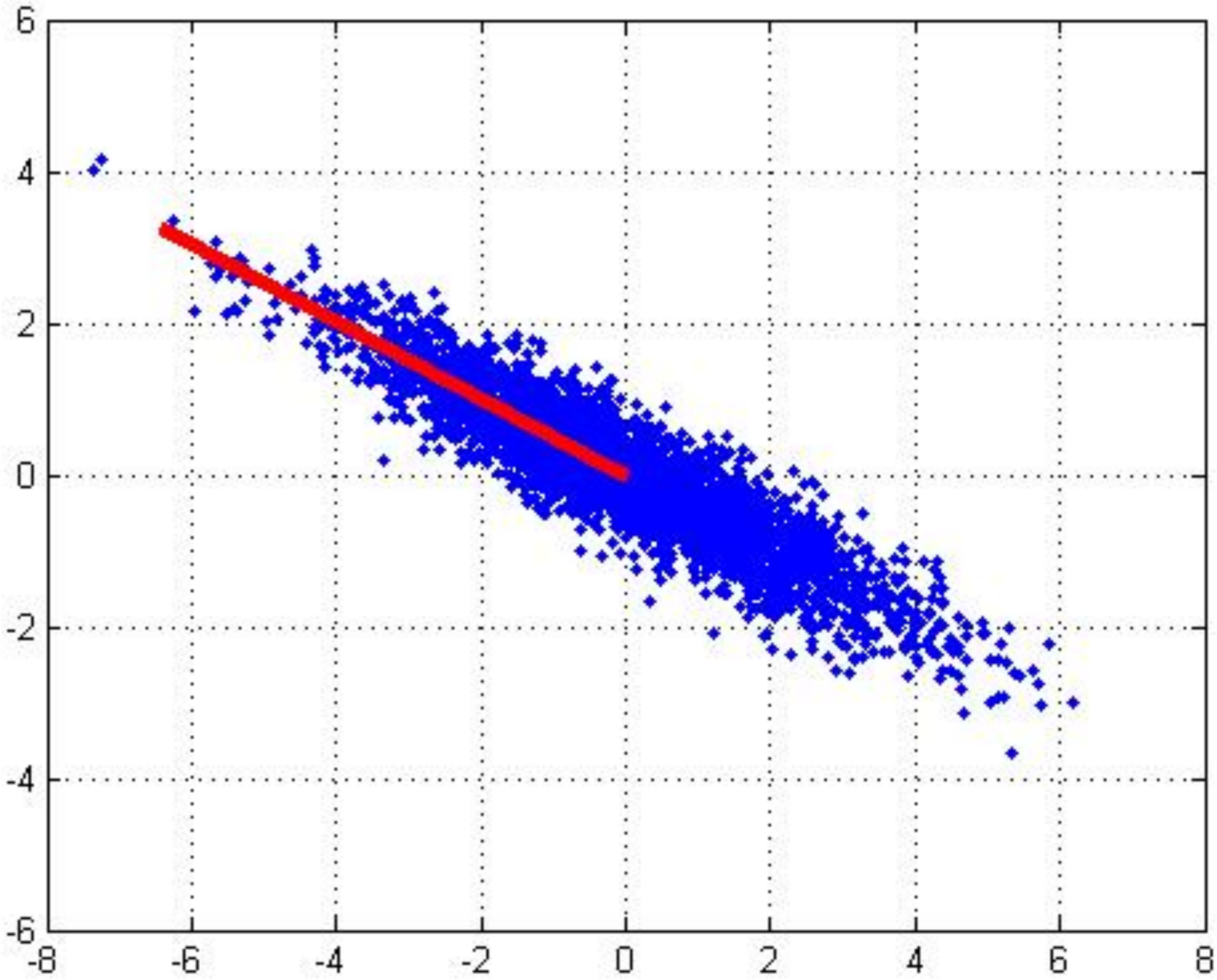
Slide from Nina Balcan

2D Gaussian dataset



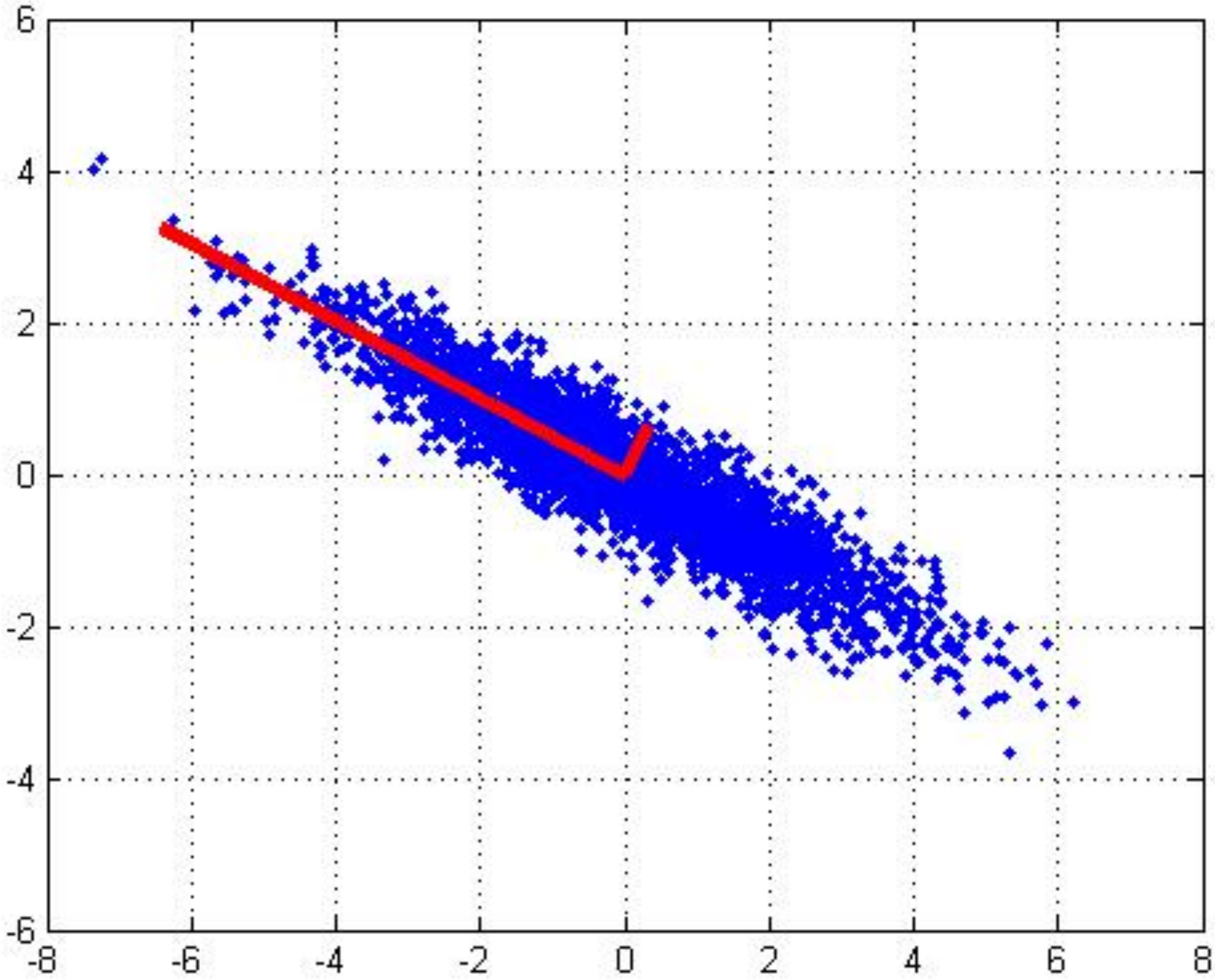
Slide from Barnabas Poczos

1st PCA axis



Slide from Barnabas Poczos

2^{nd} PCA axis



Slide from Barnabas Poczos

PCA ALGORITHMS

PCA algorithm I (sequential)

Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{(\mathbf{w}^T \mathbf{x}_i)^2\}$$

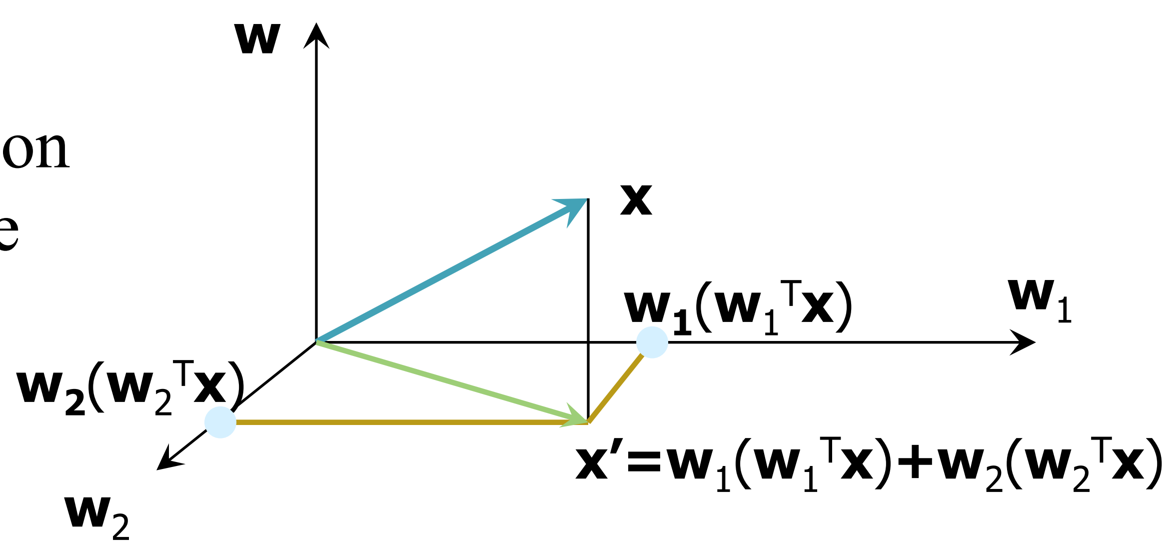
1^{st} PCA vector

We maximize the variance of projection of $\mathbf{x}$

$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i - \underbrace{\sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i}_{\mathbf{x}' \text{ PCA reconstruction}})]^2\}$$

k^{th} PCA vector

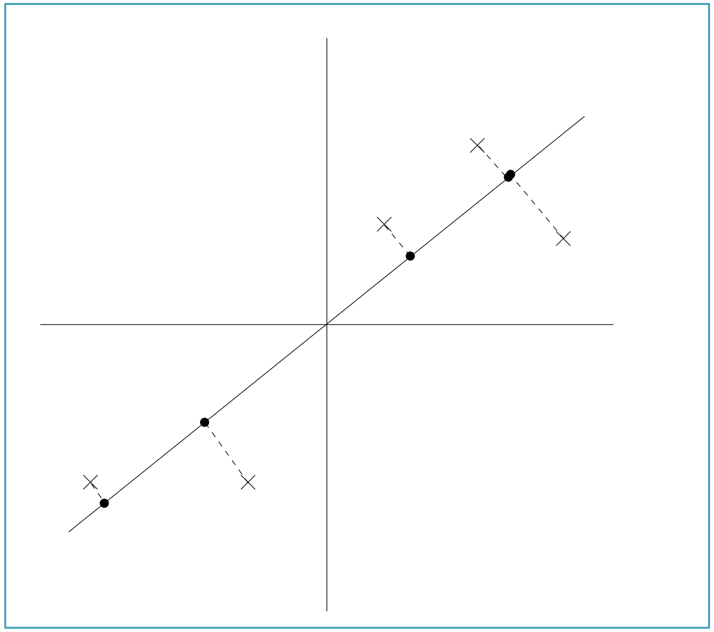
We maximize the variance of the projection in the residual subspace



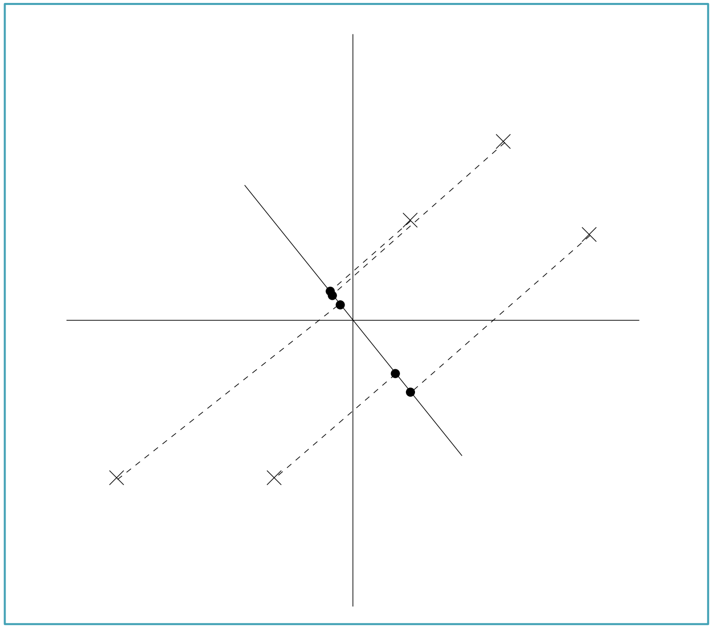
Maximizing the Variance

- Consider the two projections below
- Which maximizes the variance?

Option A



Option B



Figures from Andrew Ng (CS229 Lecture Notes)

PCA algorithm I (sequential)

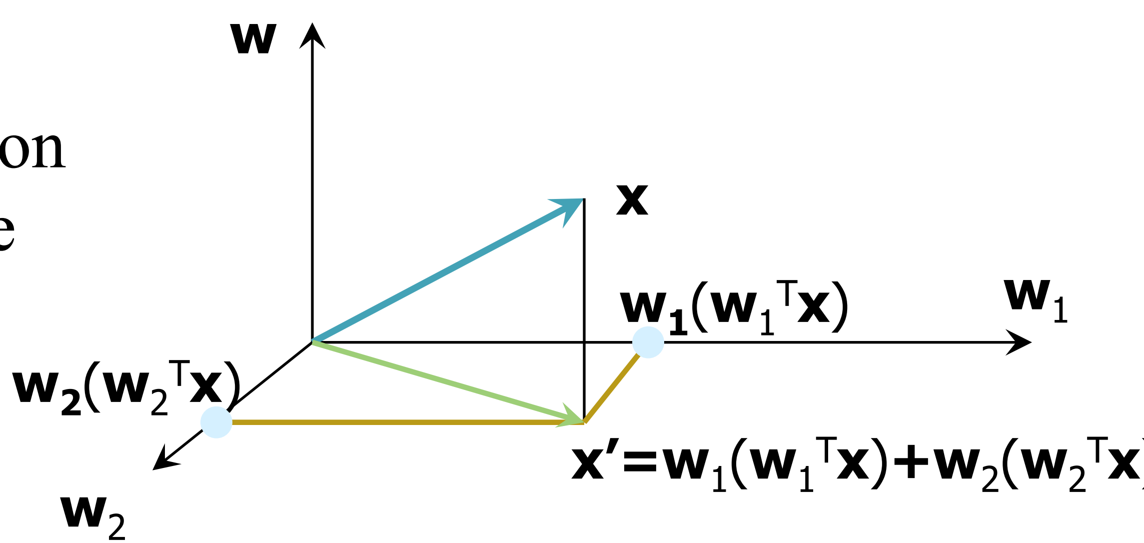
Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{(\mathbf{w}^T \mathbf{x}_i)^2\}$$ 1st PCA vector

We maximize the variance of projection of $\mathbf{x}$

$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i - \underbrace{\sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i}_{\mathbf{x}' \text{ PCA reconstruction}})]^2\}$$ kth PCA vector

We maximize the variance of the projection in the residual subspace



Slide from Barnabas Poczos

PCA algorithm II
(sample covariance matrix)

- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute covariance matrix $\boldsymbol{\Sigma}$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^T$$

where

$$\bar{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$$

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

We get the eigvectors using an eigendecomposition.
Power iteration (Von Mises iteration is a standard algorithm for this)

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

Slide from Barnabas Poczos

Why the Eigenvectors?

Maximise $\mathbf{u}^T\mathbf{X}\mathbf{X}^T\mathbf{u}$

s.t $\mathbf{u}^T\mathbf{u} = 1$

Construct Langrangian $\mathbf{u}^T\mathbf{X}\mathbf{X}^T\mathbf{u} - \lambda\mathbf{u}^T\mathbf{u}$

Vector of partial derivatives set to zero

$$\mathbf{x}\mathbf{x}^T\mathbf{u} - \lambda\mathbf{u} = (\mathbf{x}\mathbf{x}^T - \lambda\mathbf{I})\ \mathbf{u} = 0$$

As $\mathbf{u} \neq \mathbf{0}$ then $\mathbf{u}$ must be an eigenvector of $\mathbf{X}\mathbf{X}^T$ with eigenvalue λ

Eigenvalues & Eigenvectors

- For symmetric matrices, eigenvectors for distinct eigenvalues are **orthogonal**

$$Sv_{\{1,2\}} = \lambda_{\{1,2\}} v_{\{1,2\}}, \text{ and } \lambda_1 \neq \lambda_2 \Rightarrow v_1 \bullet v_2 = 0$$

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

$$\text{if } |S - \lambda I| = 0 \text{ and } \mathrm{S} = \mathrm{S}^{\mathrm{T}} \Rightarrow \lambda \in \Re$$

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

$$\forall w \in \Re^n, w^T S w \geq 0, \text{ then if } \mathrm{Sv} = \lambda \mathrm{v} \Rightarrow \lambda \geq 0$$

Eigen/diagonal 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 an **eigen decomposition**

$$\mathbf{S} = \mathbf{U} \underbrace{\mathbf{\Lambda}}_{\text{diagonal}} \mathbf{U}^{-1}$$

Unique for distinct eigen-values

(cf. matrix diagonalization theorem)

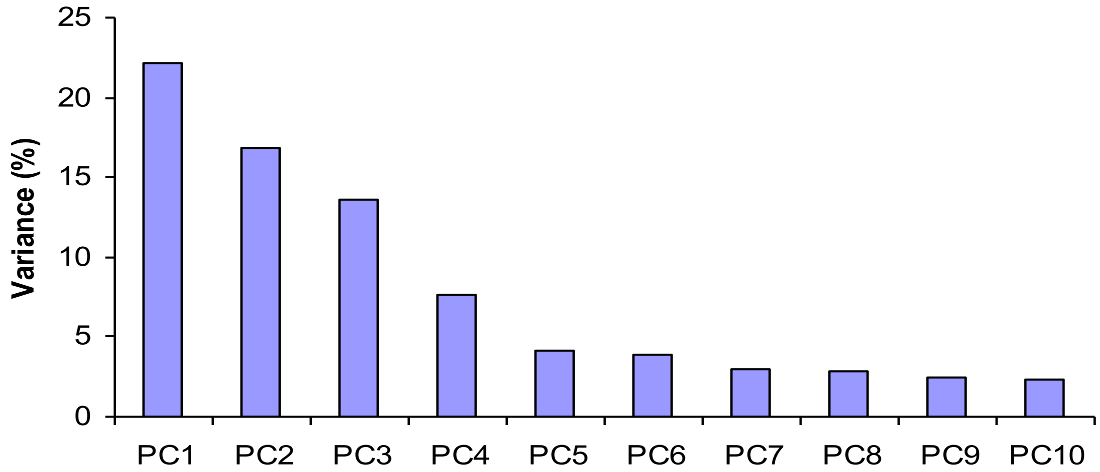
- Columns of ***U*** are **eigenvectors** of ***S***
- Diagonal elements of $\mathbf{\Lambda}$ are **eigenvalues** of $\mathbf{S}$

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

How Many PCs?

- For n original dimensions, sample covariance matrix is nxn, and has up to n eigenvectors. So n PCs.
- Where does dimensionality reduction come from?

Can *ignore* the components of lesser significance.



You do lose some information, but if the eigenvalues are small, you don't lose much

- n dimensions in original data
- calculate n eigenvectors and eigenvalues
- choose only the first p eigenvectors, based on their eigenvalues
- final data set has only p dimensions

Eigen/diagonal Decomposition

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

- **Theorem**: Exists an **eigen decomposition**

$$\mathbf{S} = \mathbf{U}\underbrace{\mathbf{\Lambda}}_{\textit{diagonal}}\mathbf{U}^{-1}$$

Unique for distinct eigen-values

(cf. matrix diagonalization theorem)

- Columns of ***U*** are **eigenvectors** of ***S***
- Diagonal elements of $\mathbf{\Lambda}$ are **eigenvalues** of $\mathbf{S}$

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

Singular Value Decomposition

For an $m \times n$ matrix $\mathbf{A}$ of rank r there exists a factorization (Singular Value Decomposition = **SVD**) as follows:

$$A = \underset{m\times m}{U}\,\underset{m\times m}{\Sigma}\,V^T \qquad V \text{ is } m\times n$$

The columns of ***U*** are orthogonal eigenvectors of AA^T.

The columns of ***V*** are orthogonal eigenvectors of A^TA.

Eigenvalues $\lambda_1 \ldots \lambda_r$ of AA^T are the eigenvalues of A^TA.

$$\sigma_i = \sqrt{\lambda_i}$$

$$\Sigma = diag(\sigma_1 \ldots \sigma_r)$$

Singular values.

PCA: Two Interpretations

E.g., for the first component.

Maximum Variance Direction: 1st PC a vector v such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections)

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^T\mathbf{x}_i)^2 = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v}$$

Minimum Reconstruction Error: 1st PC a vector v such that projection on to this vector yields minimum MSE reconstruction

$$\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - (\mathbf{v}^T\mathbf{x}_i)\mathbf{v}\|^2$$

PCA: Two Interpretations

E.g., for the first component.

Maximum Variance Direction: 1st PC a vector v such that projection on to this vector capture maximum variance in the data (out of all possible one dimensional projections)

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^T\mathbf{x}_i)^2 = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v}$$

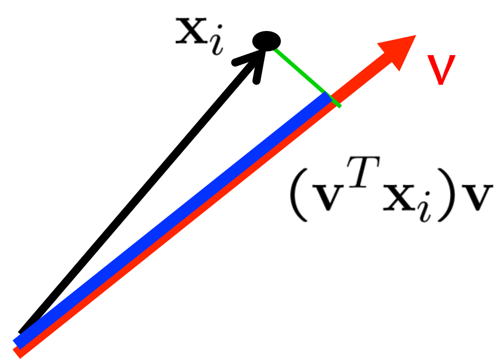
Minimum Reconstruction Error: 1st PC a vector v such that projection on to this vector yields minimum MSE reconstruction

$$\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - (\mathbf{v}^T\mathbf{x}_i)\mathbf{v}\|^2$$

blue² + green² = black²

black² is fixed (it's just the data)

So, maximizing blue² is equivalent to minimizing green²



Slide from Nina Balcan

PCA algorithm III (SVD of the data matrix)

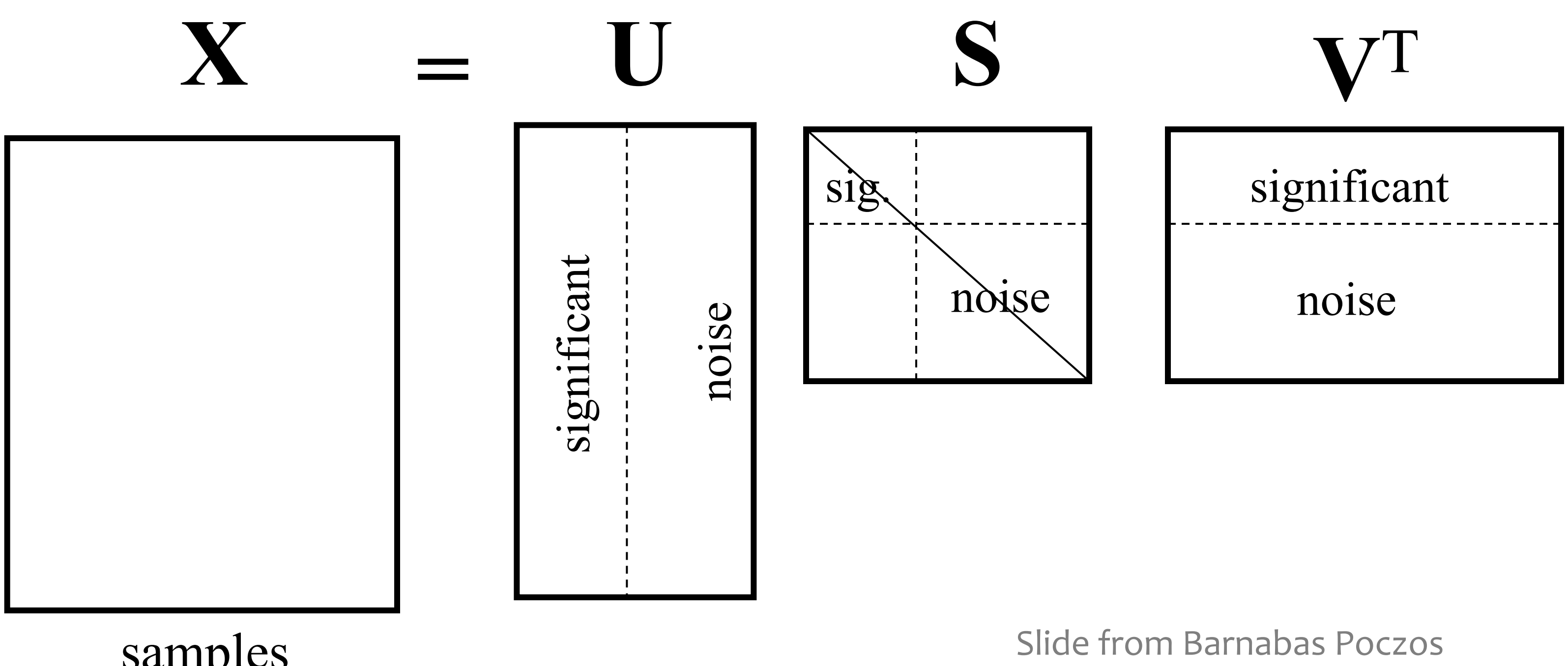
Singular Value Decomposition of the **centered** data matrix **X**.

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m] \in \mathbb{R}^{N \times m},$$

m: number of instances,

N: dimension

$$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{USV}^{\mathrm{T}}$$



Slide from Barnabas Poczos

PCA algorithm III

- **Columns of U**
  - the principal vectors, $\{ \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)} \}$
  - orthogonal and has unit norm – so $U^T U = I$
  - Can reconstruct the data using linear combinations of $\{ \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)} \}$

- **Matrix S**
  - Diagonal
  - Shows importance of each eigenvector

- **Columns of V^T**
  - The coefficients for reconstructing the samples

Slide from Barnabas Poczos

SVD and PCA

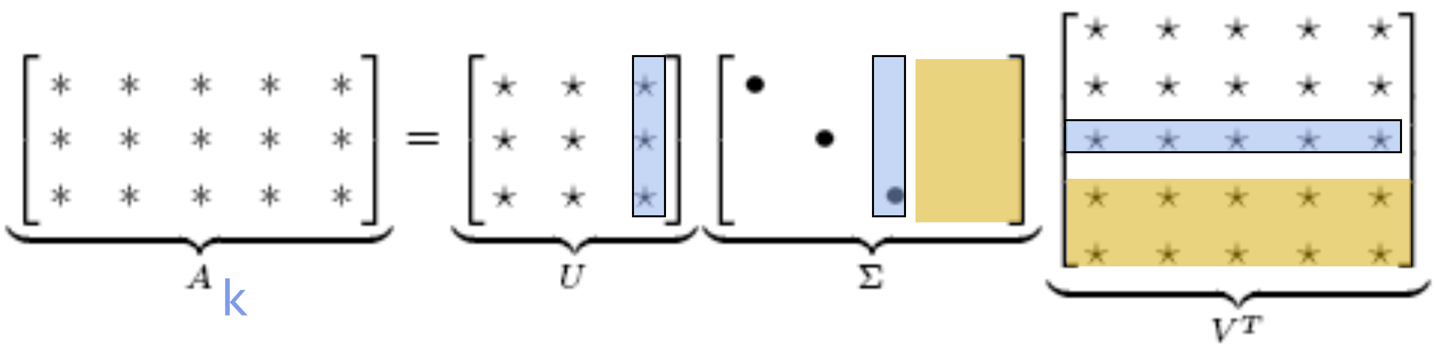
- The first root is called the prinicipal eigenvalue which has an associated orthonormal ($\mathbf{u}^T\mathbf{u} = 1$) *eigenvector* $\mathbf{u}$
- Subsequent roots are ordered such that $\lambda_1 > \lambda_2 > \ldots > \lambda_M$ with rank(**D**) non-zero values.
- Eigenvectors form an orthonormal basis i.e. $\mathbf{u}_i^T\mathbf{u}_j = \delta_{ij}$
- The eigenvalue decomposition of $\mathbf{XX}^T = \mathbf{U\Sigma U}^T$
- where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M]$ and $\mathbf{\Sigma} = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_M]$
- Similarly the eigenvalue decomposition of $\mathbf{X}^T\mathbf{X} = \mathbf{V\Sigma V}^T$
- The SVD is closely related to the above $\mathbf{X}=\mathbf{U}\,\mathbf{\Sigma}^{1/2}\,\mathbf{V}^T$
- The left eigenvectors **U**, right eigenvectors **V**,
- singular values = square root of eigenvalues.

Low-rank Approximation

- Solution via SVD

$$A_k = U \operatorname{diag}(\sigma_1, \ldots, \sigma_k, \underbrace{0, \ldots, 0}) V^T$$

set smallest r-k
singular values to zero



$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

← *column notation: sum of rank 1 matrices*

Approximation error

- How good (bad) is this approximation?
- It's the best possible, measured by the Frobenius norm of the error:

$$\min_{X:rank(X)=k} \|A - X\|_F = \|A - A_k\|_F = \sigma_{k+1}$$

where the σ_i are ordered such that $\sigma_i \geq \sigma_{i+1}$.

Suggests why Frobenius error drops as *k* increased.

Slides from Barnabas Poczos

Original sources include:

- Karl Booksh Research group
- Tom Mitchell
- Ron Parr

PCA EXAMPLES

Face recognition

Challenge: Facial Recognition

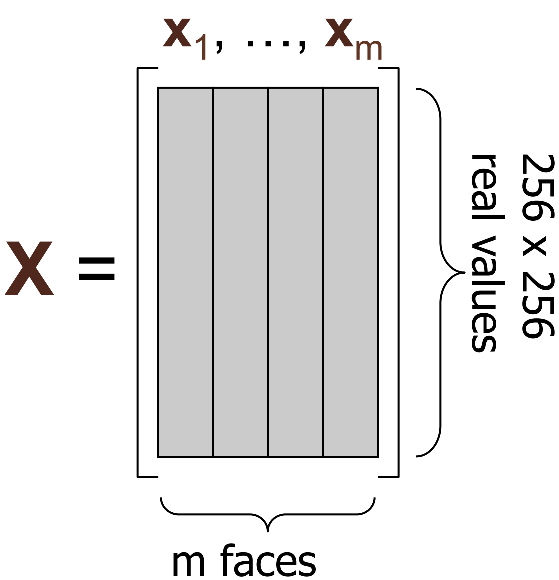
- Want to identify specific person, based on facial image
- Robust to glasses, lighting,…

⇒ Can't just use the given 256 x 256 pixels

Applying PCA: Eigenfaces

Method A: Build a PCA subspace for each person and check which subspace can reconstruct the test image the best

Method B: 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 $\mathbf{x}$ is …
- 256×256 values (luminance at location)
  - $\mathbf{x}$ in $\Re^{256\times256}$ (view as 64K dim vector)
- Form $\mathbf{X} = [\ \mathbf{x}_1, \ldots, \mathbf{x}_m\ ]$ **centered** data mtx
- Compute $\Sigma = \mathbf{XX}^T$
- Problem: Σ is $64K \times 64K$ … HUGE!!!

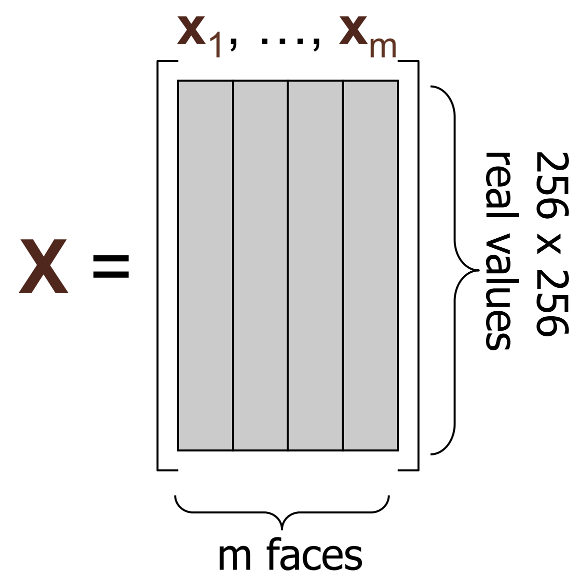
Computational Complexity

- Suppose m instances, each of size N
  - Eigenfaces: $m=500$ faces, each of size $N=64K$
- Given $N \times N$ covariance matrix Σ, can compute
  - all N eigenvectors/eigenvalues in $O(N^3)$
  - first k eigenvectors/eigenvalues in $O(k\ N^2)$

- But if $N=64K$, EXPENSIVE!

A Clever Workaround

- Note that $m << 64K$
- Use $\mathbf{L} = \mathbf{X}^T\mathbf{X}$ instead of $\mathbf{\Sigma} = \mathbf{X}\mathbf{X}^T$
- If $\mathbf{v}$ is eigenvector of $\mathbf{L}$
  then $\mathbf{Xv}$ is eigenvector of $\mathbf{\Sigma}$



Proof:

$$\mathbf{L}\ \mathbf{v} = \gamma\ \mathbf{v}$$

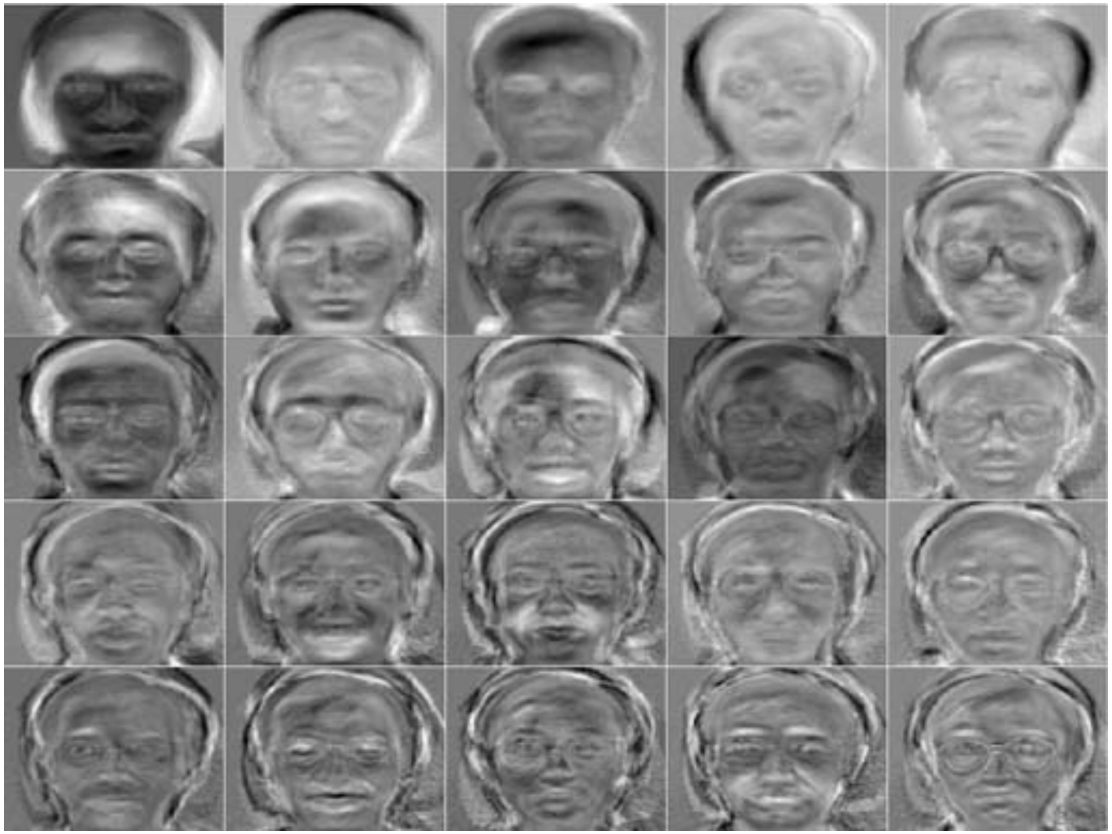
$$\mathbf{X}^T\mathbf{X}\ \mathbf{v} = \gamma\ \mathbf{v}$$

$$\mathbf{X}\ (\mathbf{X}^T\mathbf{X}\ \mathbf{v}) = \mathbf{X}(\gamma\ \mathbf{v}) = \gamma\ \mathbf{Xv}$$

$$(\mathbf{X}\mathbf{X}^T)\mathbf{X}\ \mathbf{v} = \gamma\ (\mathbf{Xv})$$

$$\mathbf{\Sigma}\ (\mathbf{Xv}) = \gamma\ (\mathbf{Xv})$$

Principle Components (Method B)



Reconstructing… (Method B)

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

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

Facial expression recognition

Happiness subspace (method A)

Disgust subspace (method A)

Facial Expression Recognition Movies (method A)

Facial Expression Recognition Movies (method A)

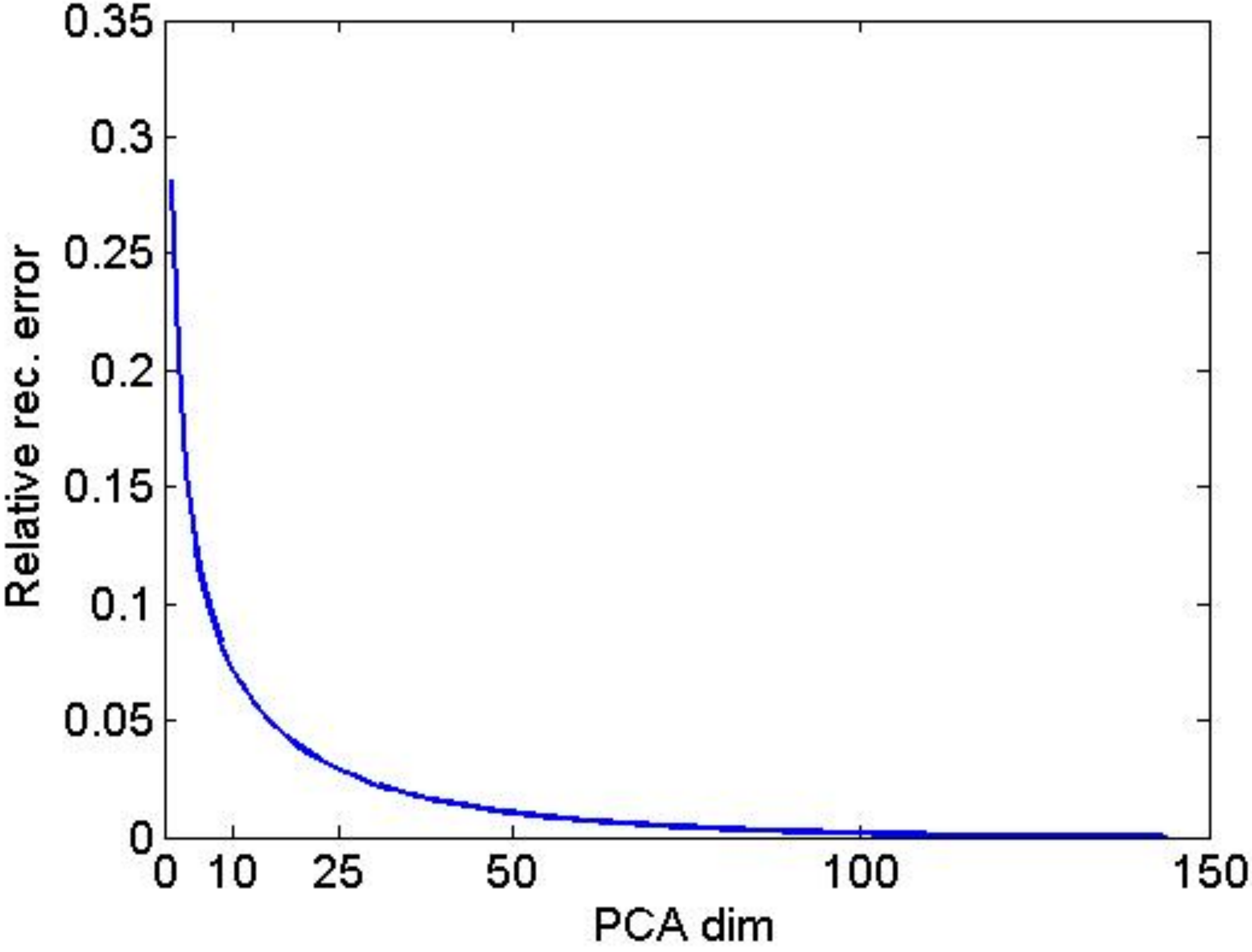
Facial Expression Recognition Movies (method A)

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

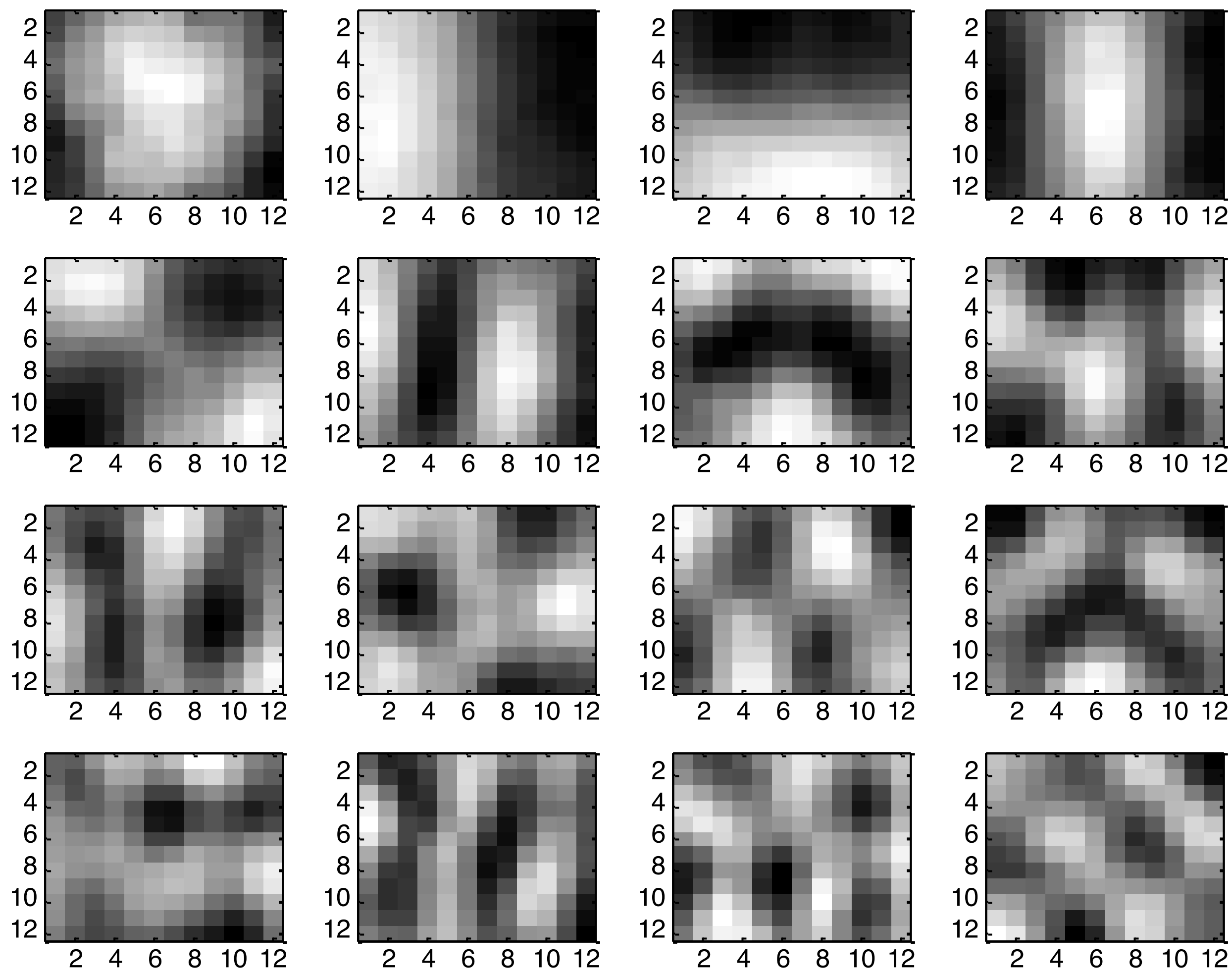
L_2 error and PCA dim



PCA compression: 144D) 60D

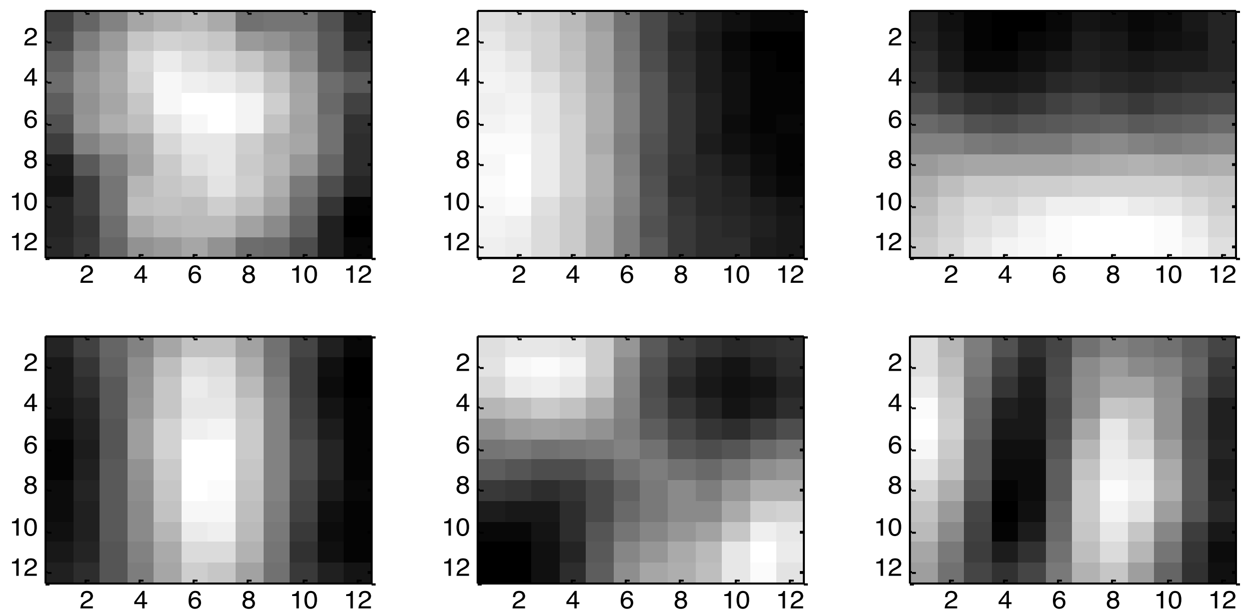
PCA compression: 144D) 16D

16 most important eigenvectors



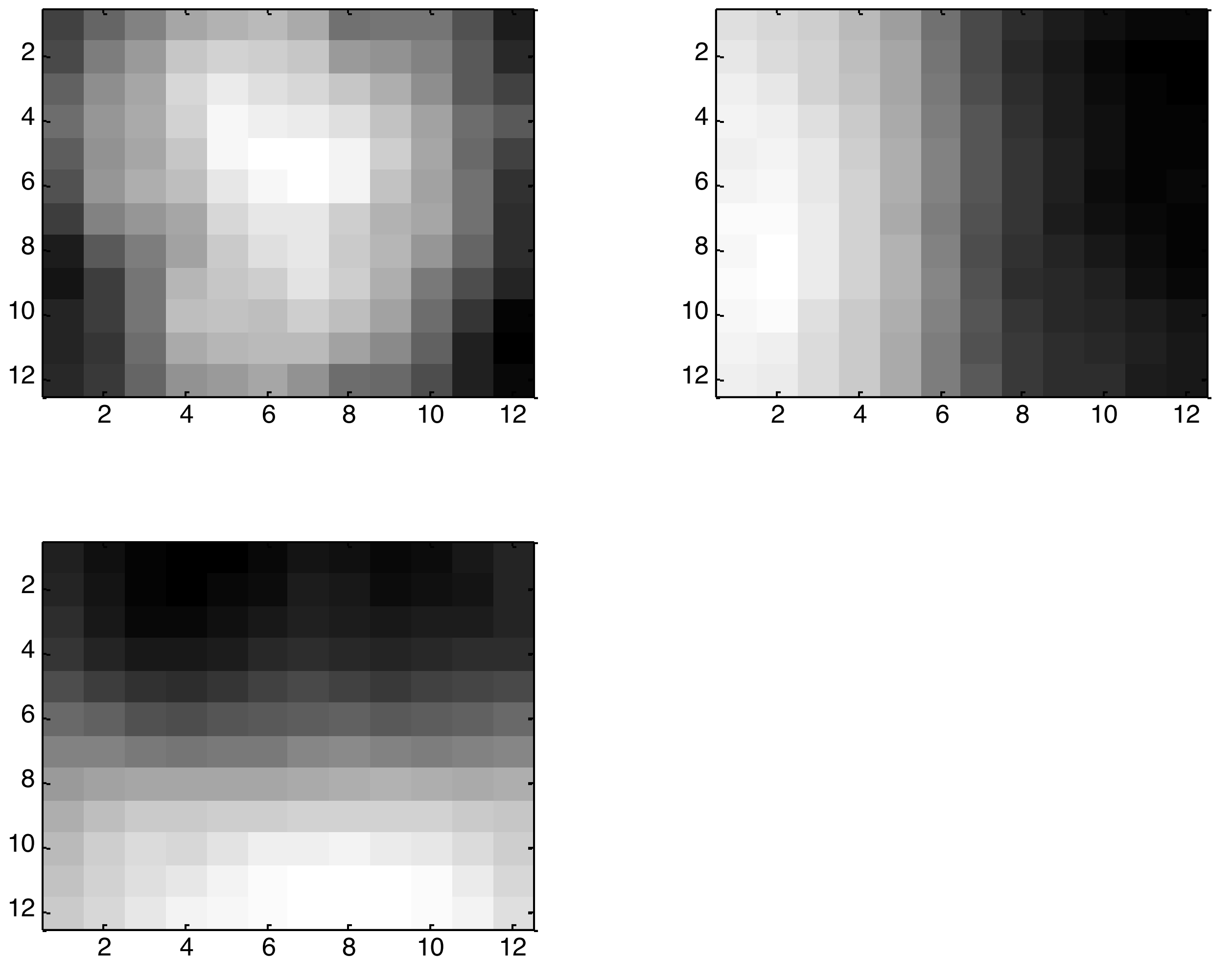
PCA compression: 144D) 6D

6 most important eigenvectors

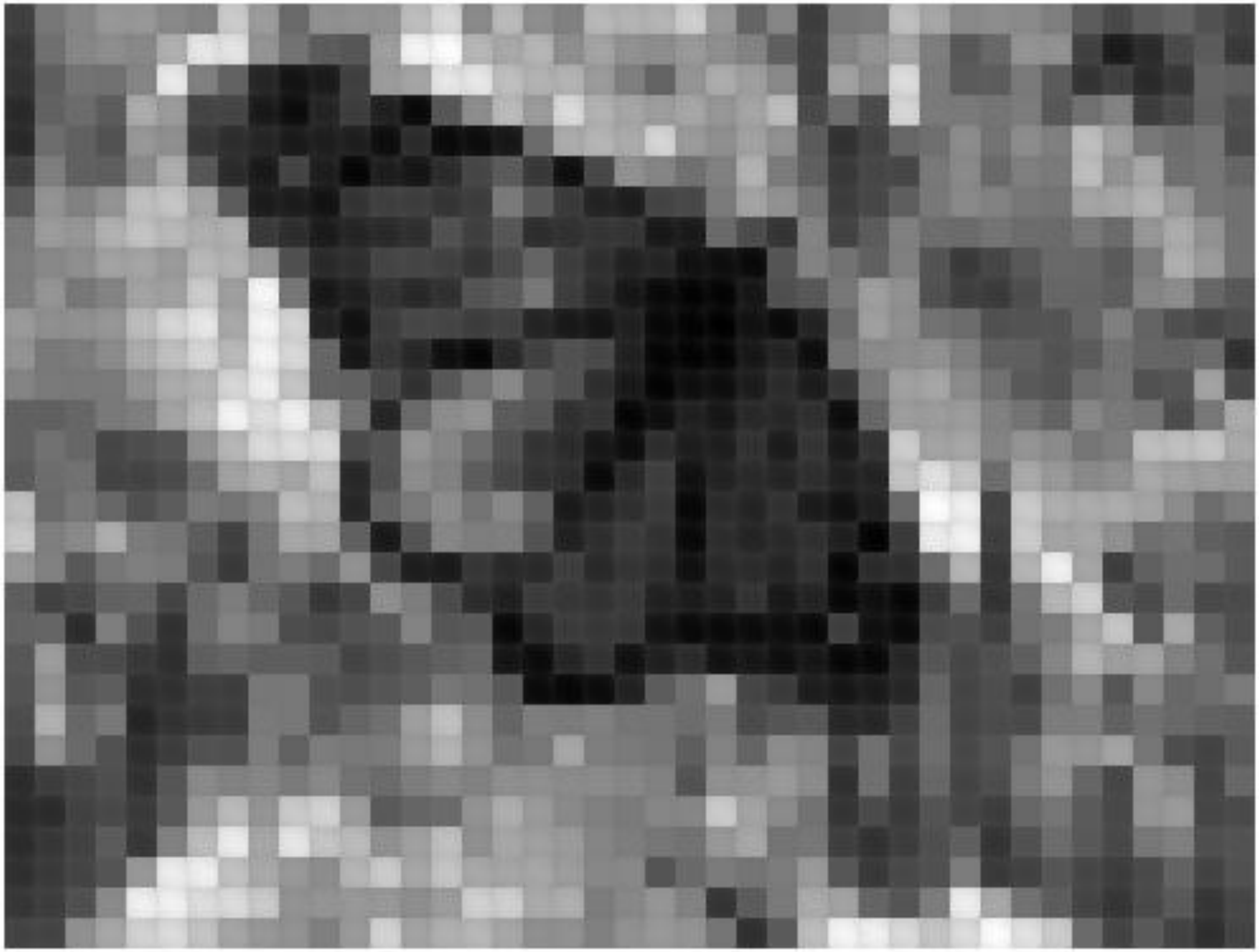


PCA compression: 144D) 3D

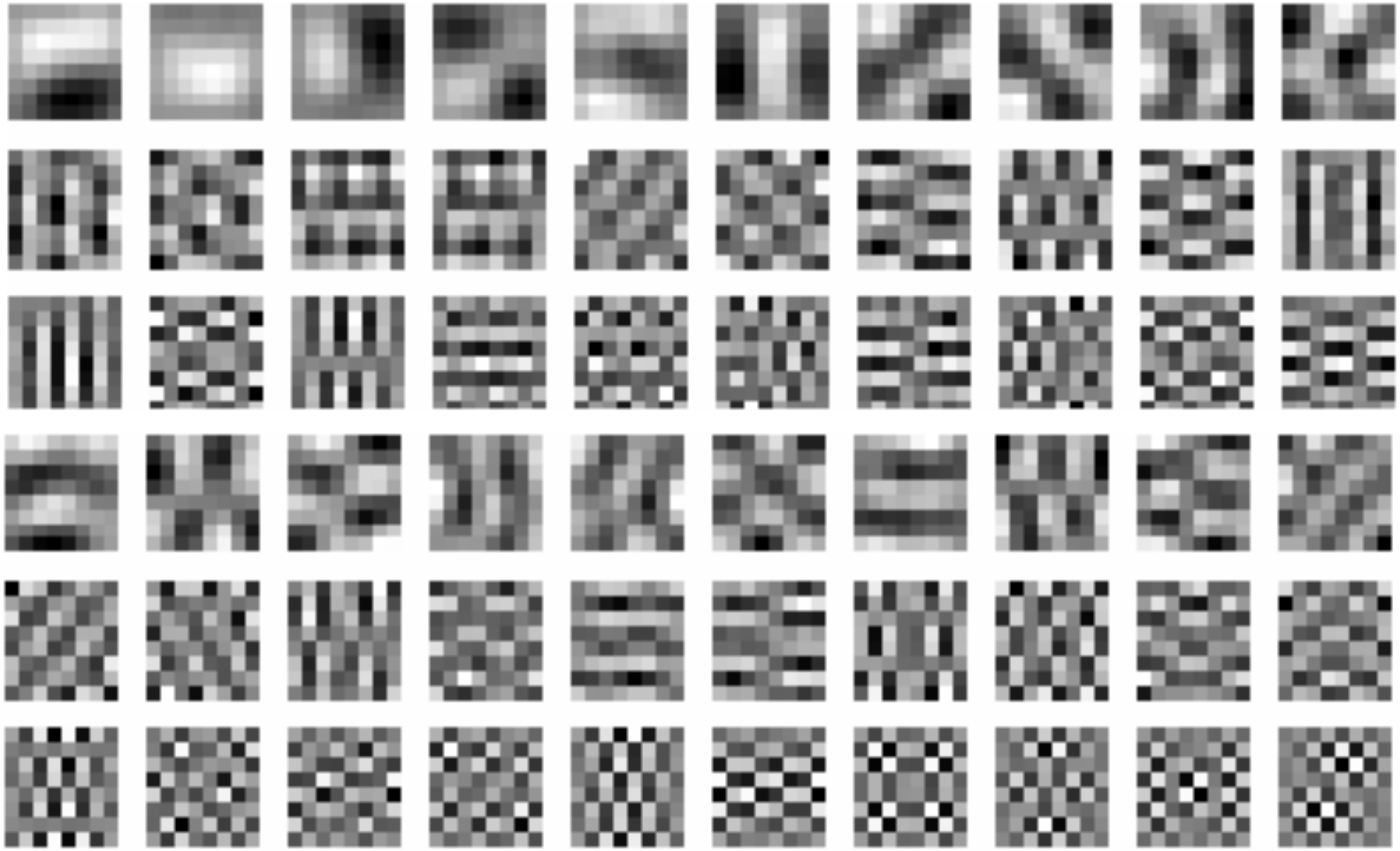
3 most important eigenvectors



PCA compression: 144D) 1D

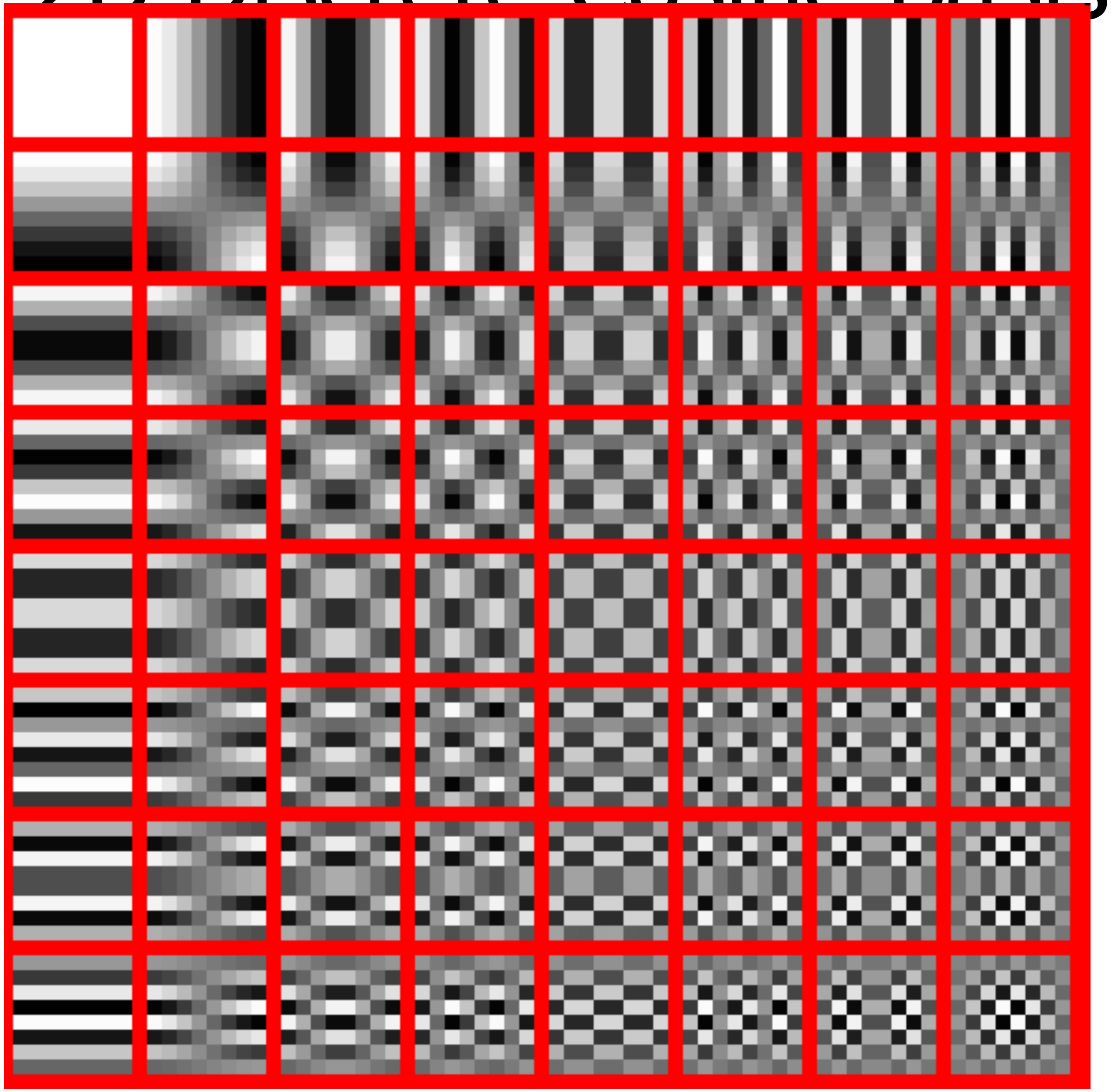


60 most important eigenvectors



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

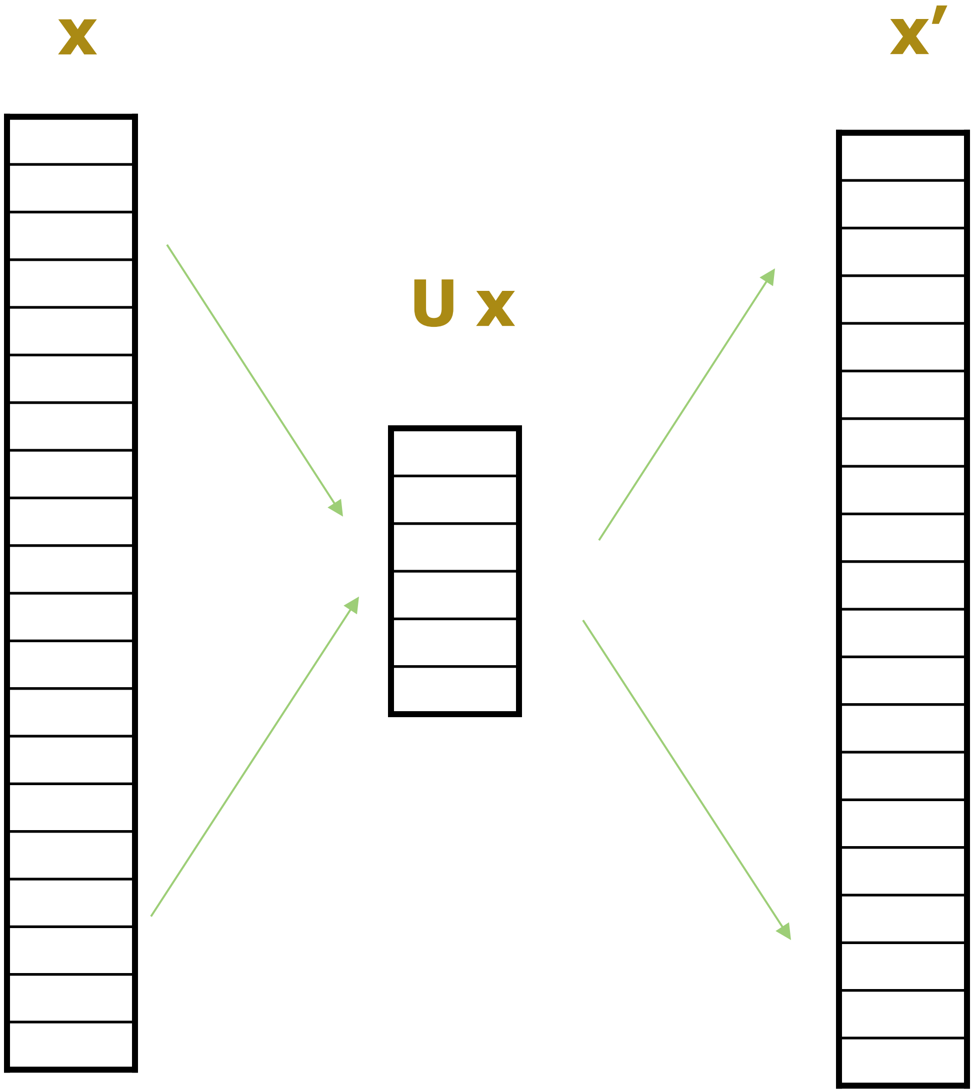
2D Discrete Cosine Basis



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

Noise Filtering

Noise Filtering, Auto-Encoder…



Noisy image

Denoised image
using 15 PCA components