



BBM406

Fundamentals of Machine Learning

Lecture 23: Dimensionality Reduction

Administrative

Project Presentations

January 8,10, 2020

- Each project group will have ~8 mins to present their work in class. The suggested outline for the presentations are as follows:
 - High-level overview of the paper (main contributions)
 - Problem statement and motivation (clear definition of the problem, why it is interesting and important)
 - Key technical ideas (overview of the approach)
 - Experimental set-up (datasets, evaluation metrics, applications)
 - Strengths and weaknesses (discussion of the results obtained)
- In addition to classroom presentations, each group should also prepare an engaging video presentation of their work using online tools such as PowToon, moovly or GoAnimate (**due January 12, 2020**).

Final Reports (Due January 15, 2019)

- The report should be prepared using LaTeX and 6-8 pages. A typical organization of a report might follow:
 - Title, Author(s).
 - Abstract. This section introduces the problem that you investigated by providing a general motivation and briefly discusses the approach(es) that you explored.
 - Introduction.
 - Related Work. This section discusses relevant literature for your project topic.
 - The Approach. This section gives the technical details about your project work. You should describe the representation(s) and the algorithm(s) that you employed or proposed as detailed and specific as possible.
 - Experimental Results. This section presents some experiments in which you analyze the performance of the approach(es) you proposed or explored. You should provide a qualitative and/or quantitative analysis, and comment on your findings. You may also demonstrate the limitations of the approach(es).
 - Conclusions. This section summarizes all your project work, focusing on the key results you obtained. You may also suggest possible directions for future work.
 - References. This section gives a list of all related work you reviewed or used

Last time... Graph-Theoretic Clustering

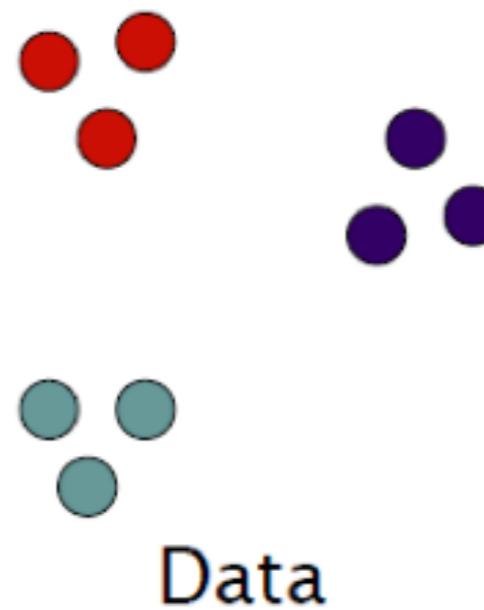
Goal: Given data points X_1, \dots, X_n and similarities $W(X_i, X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: $G(V, E, W)$

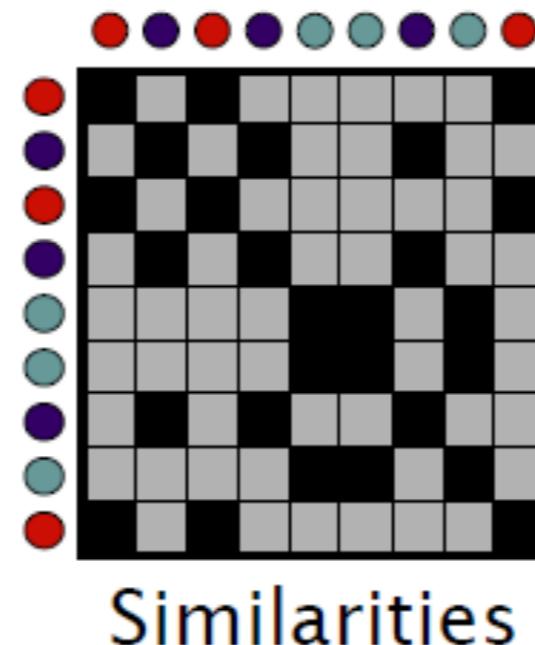
V – Vertices (Data points)

E – Edge if similarity > 0

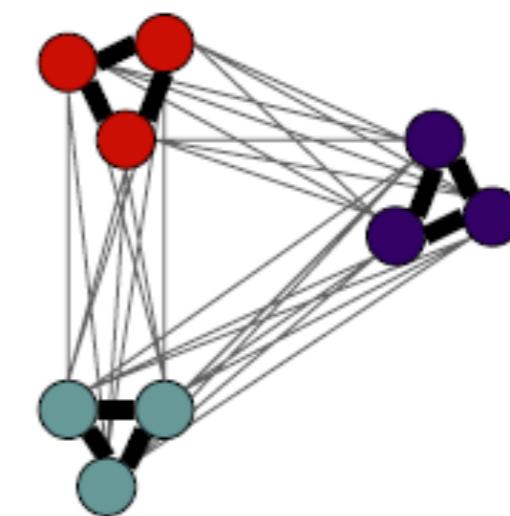
W - Edge weights (similarities)



Data



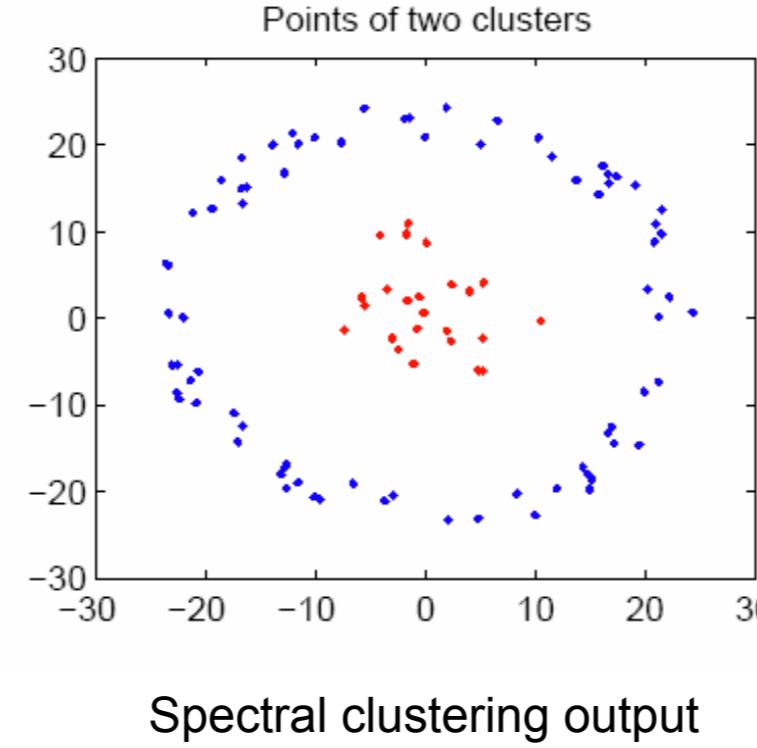
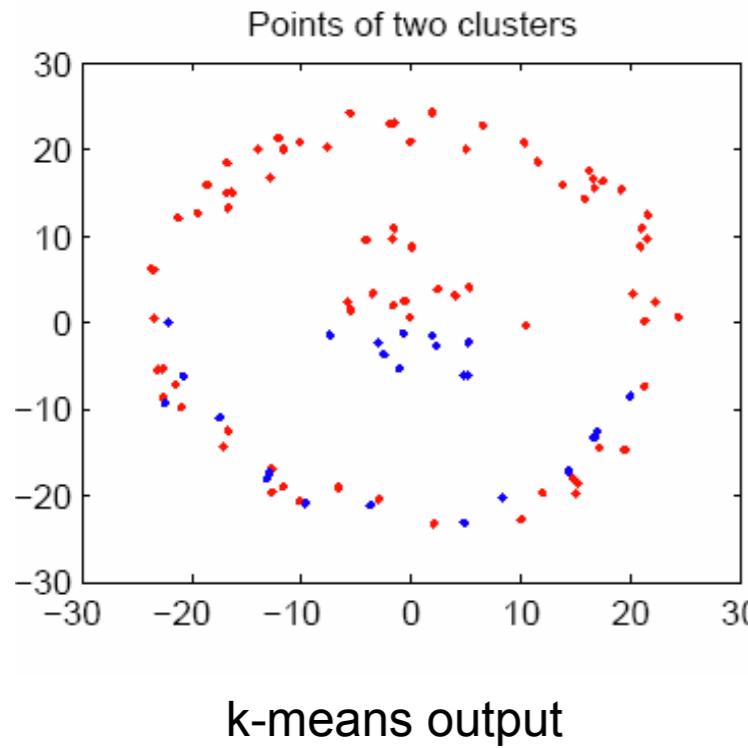
Similarities



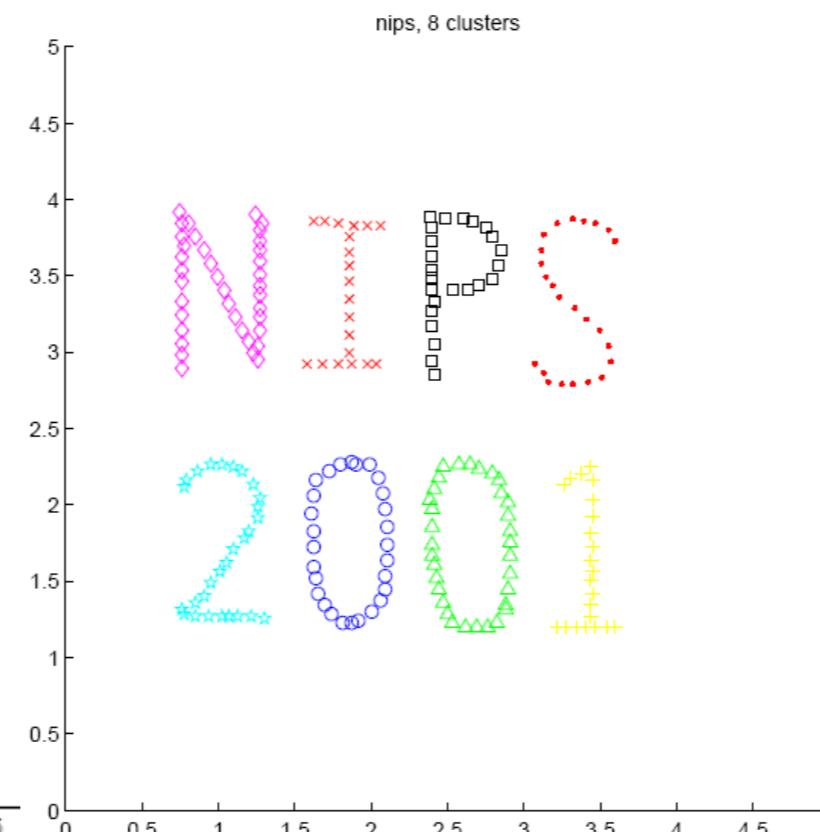
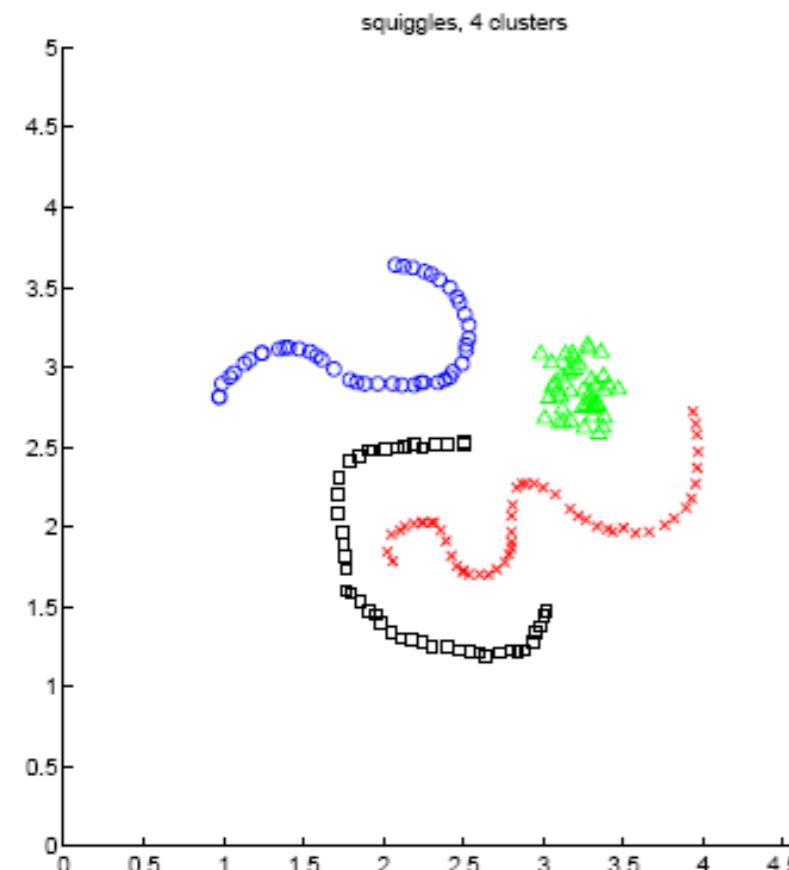
Similarity graph

Partition the graph so that edges within a group have large weights and edges across groups have small weights.

Last time... K-Means vs. Spectral Clustering



- Applying k-means to Laplacian eigenvectors allows us to **find cluster with non-convex boundaries.**

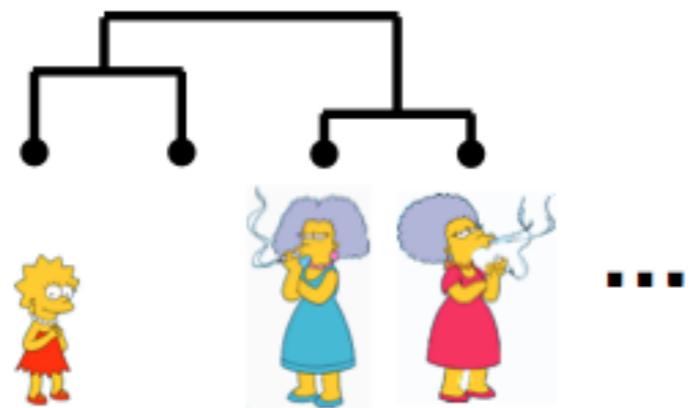


Last time...

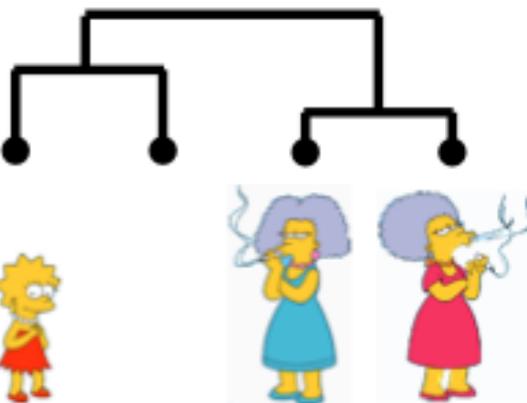
Bottom-Up (agglomerative):

Start with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

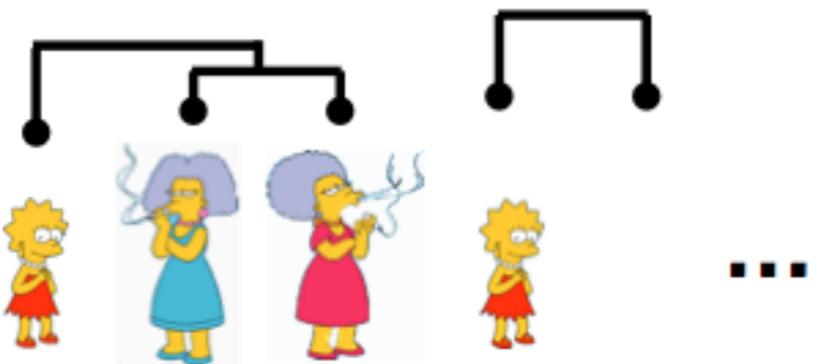
Consider all possible merges...



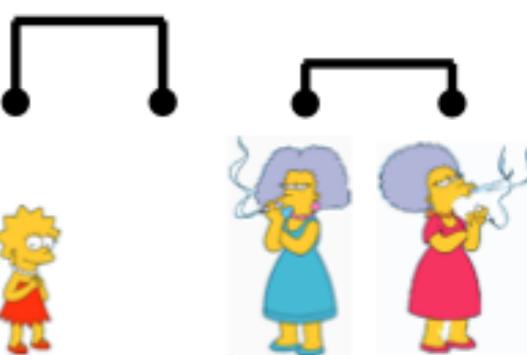
Choose the best



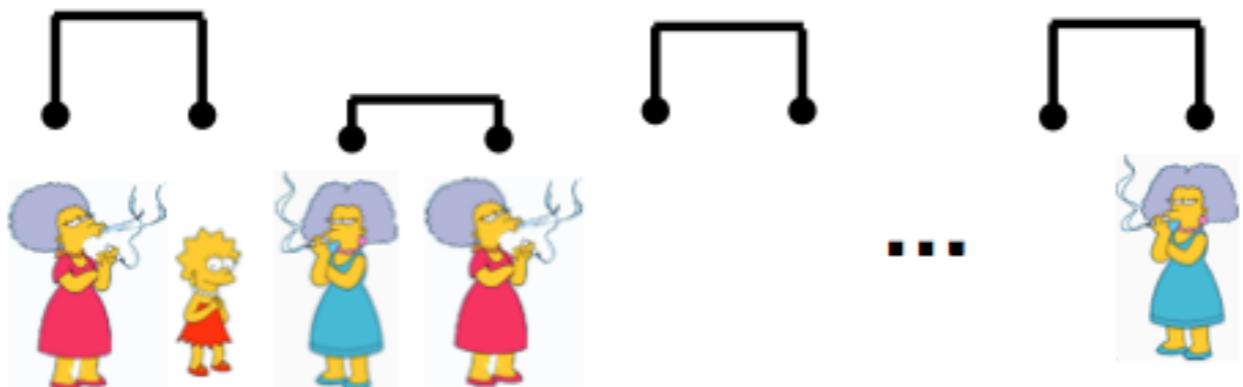
Consider all possible merges...



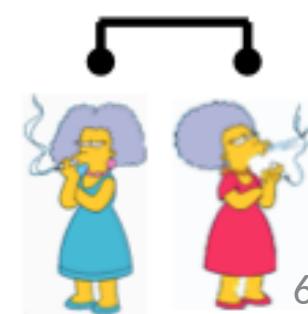
Choose the best



Consider all possible merges...



Choose the best



Today

- Dimensionality Reduction
- Principle Component Analysis (PCA)
- PCA Applications
- PCA Shortcomings
- Autoencoders
- Independent Component Analysis

Dimensionality Reduction

Motivation I: Data Visualization

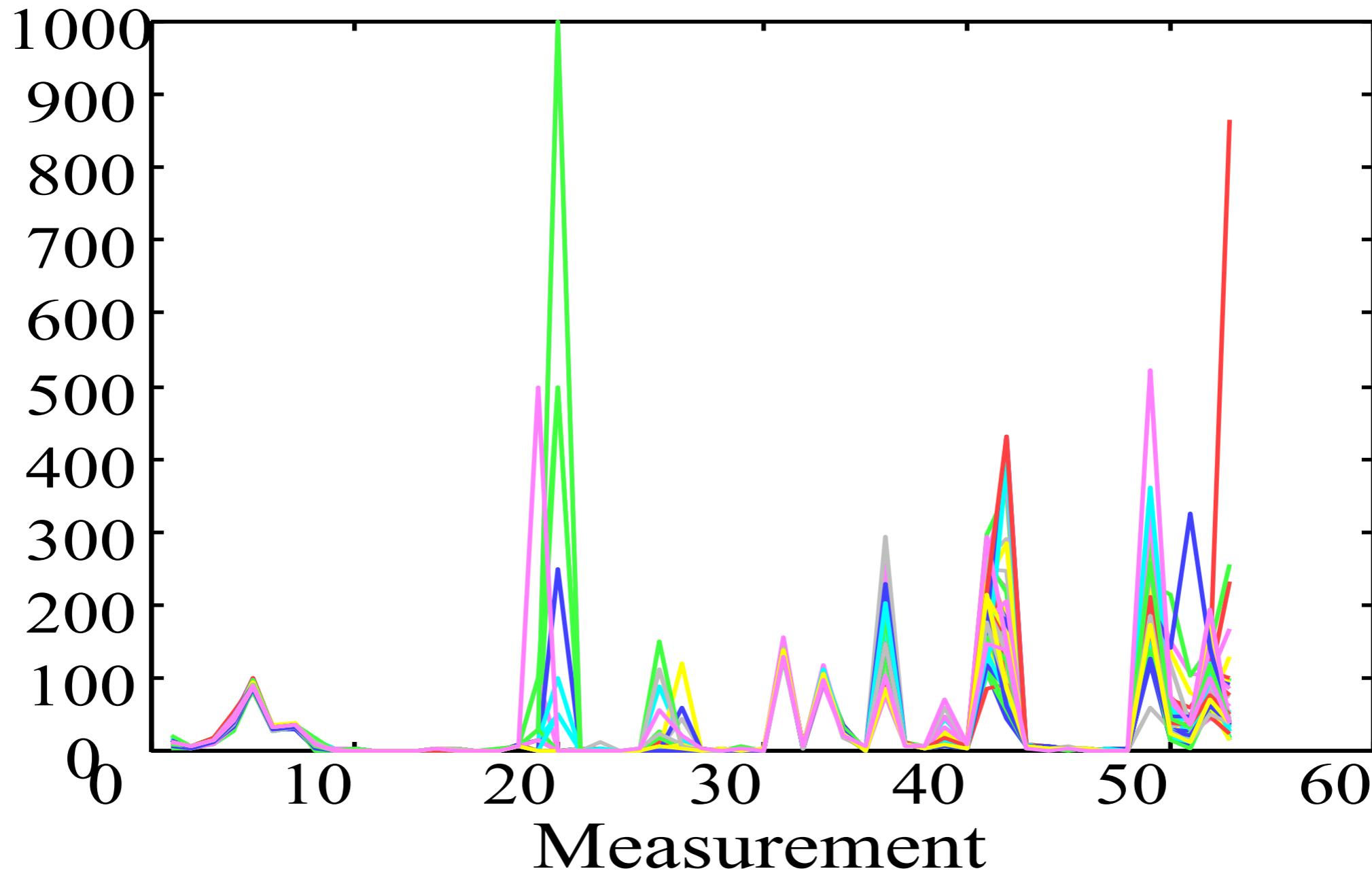
Instances

	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	84.0000	28.0000	33.0000
A6	6.9000	4.8600	16.0000	47.0000	97.0000	33.0000	34.0000
A7	7.8000	4.6800	14.7000	43.0000	92.0000	31.0000	34.0000
A8	8.6000	4.8200	15.8000	42.0000	88.0000	33.0000	37.0000
A9	5.1000	4.7100	14.0000	43.0000	92.0000	30.0000	32.0000

Features

- 53 Blood and urine samples from 65 people
- Difficult to see the correlations between features

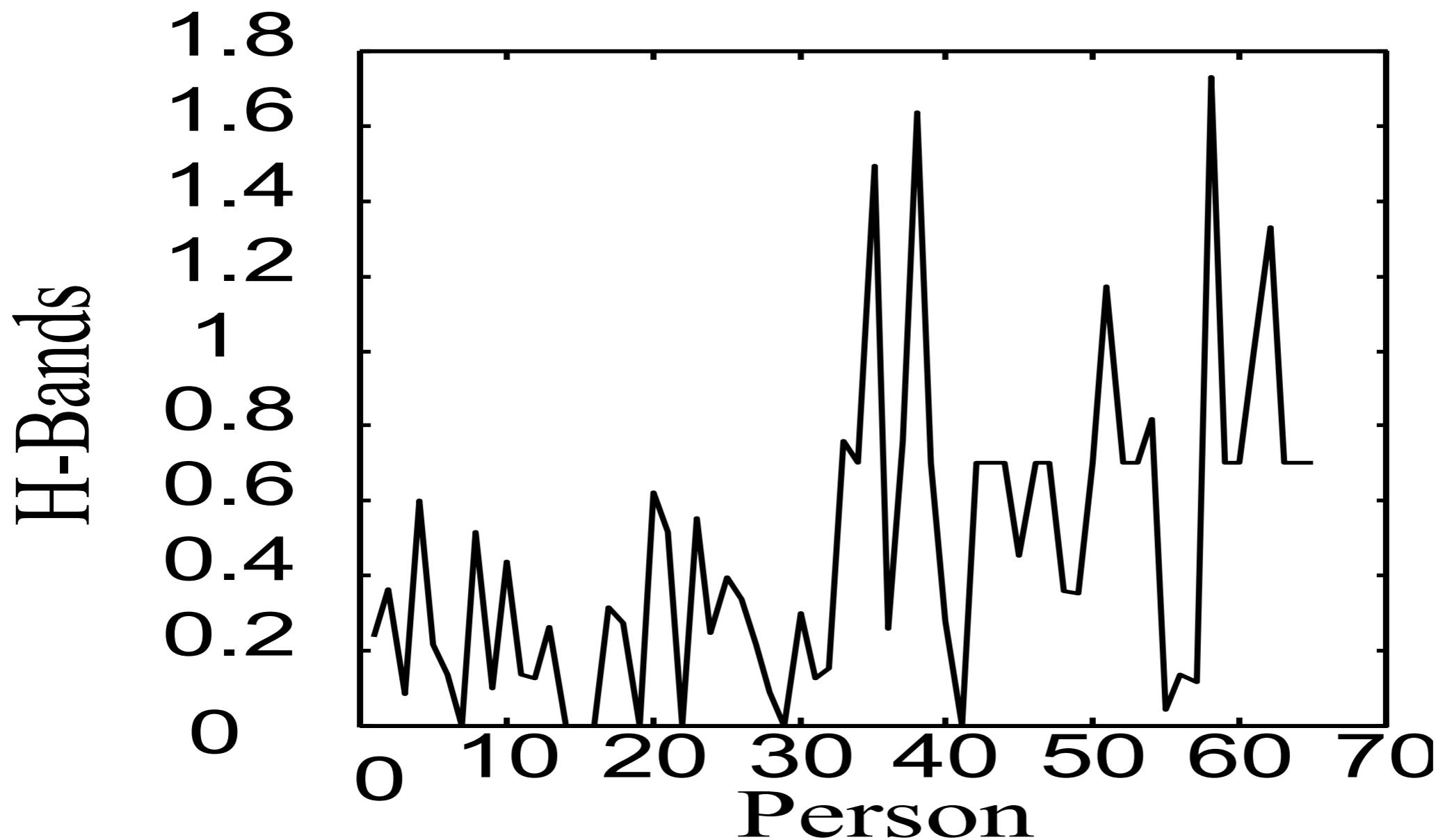
Motivation I: Data Visualization



- Spectral format (65 curves, one for each person)
- Difficult to compare different patients

Motivation I: Data Visualization

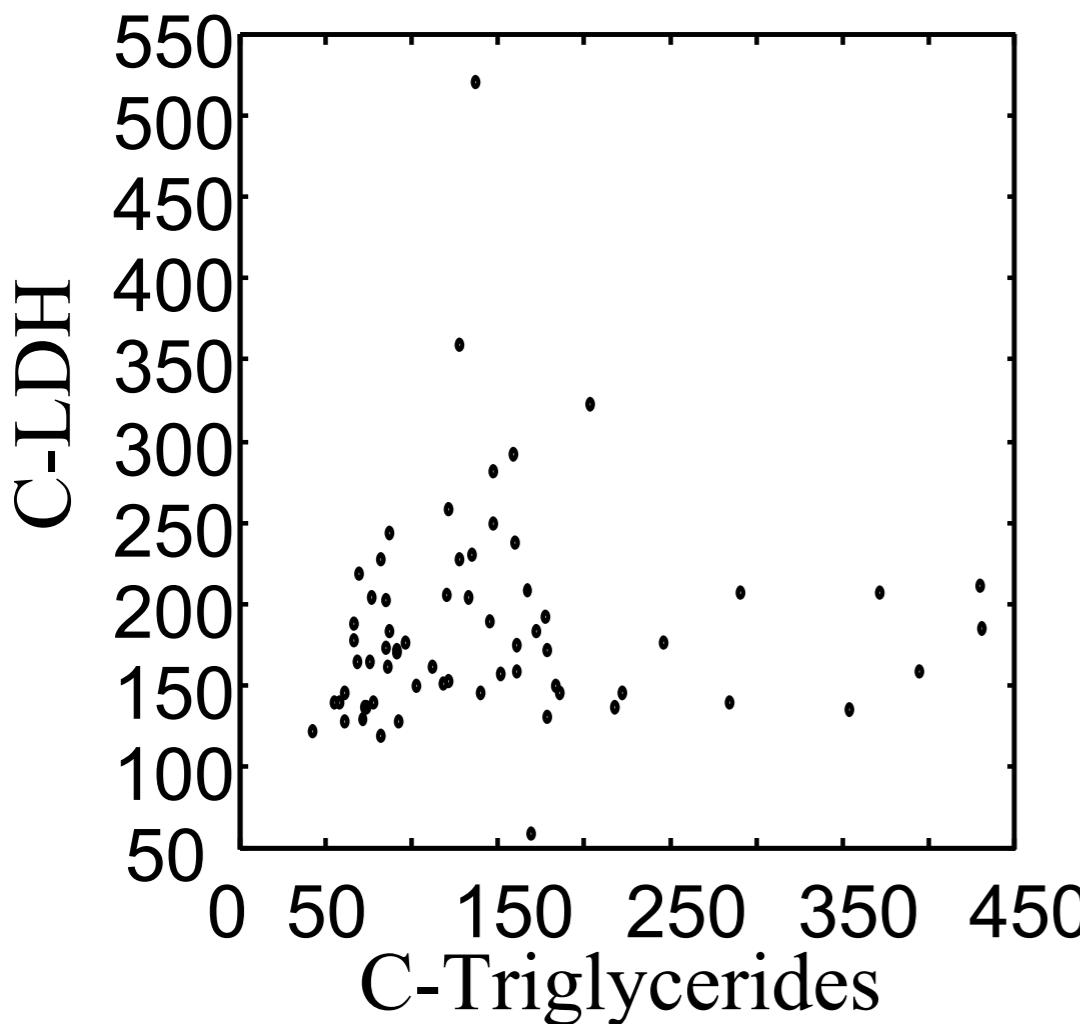
- Spectral format (53 pictures, one for each feature)



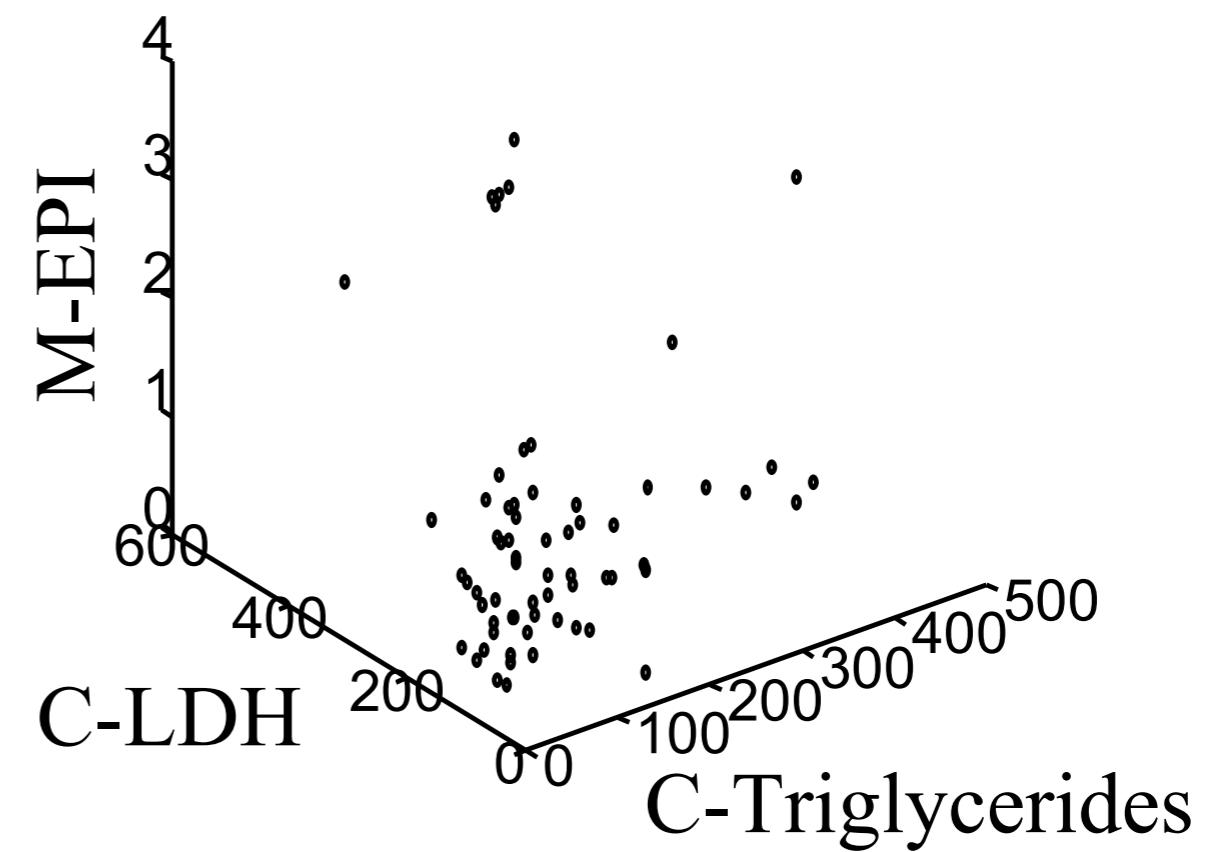
- Difficult to see the correlations between features

Motivation I: Data Visualization

Bi-variate



Tri-variate

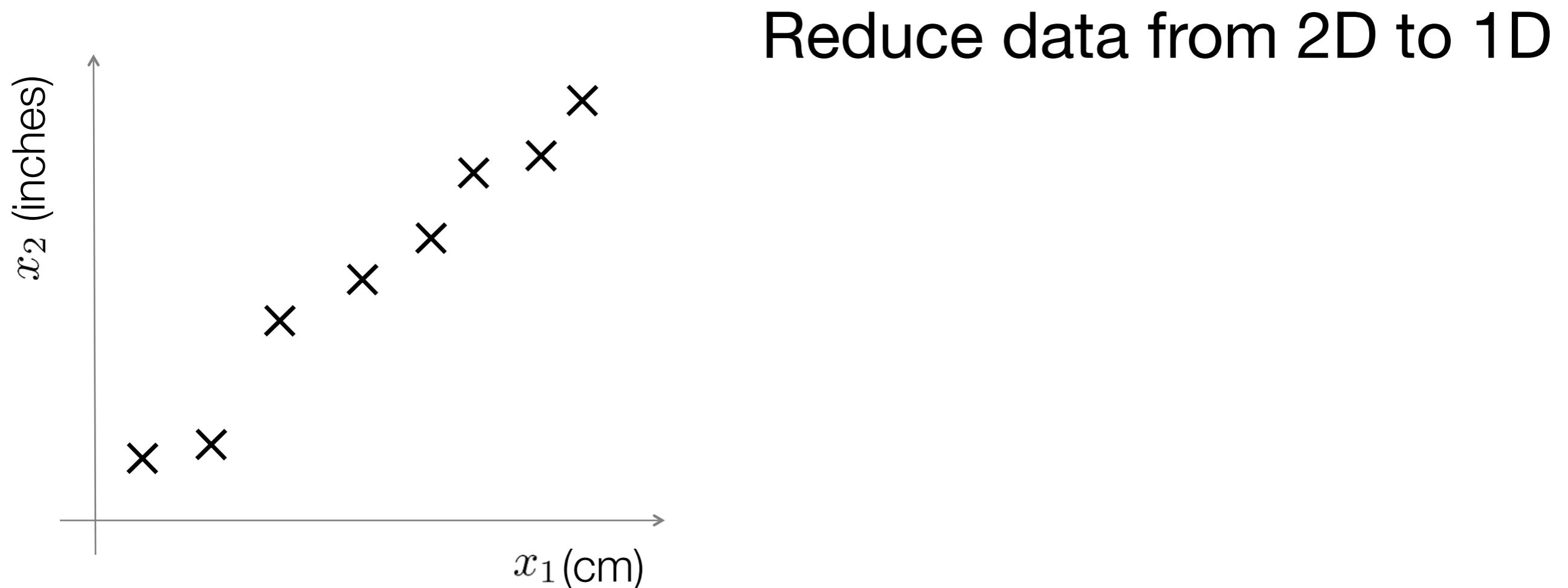


Even 3 dimensions are already difficult. How to extend this?

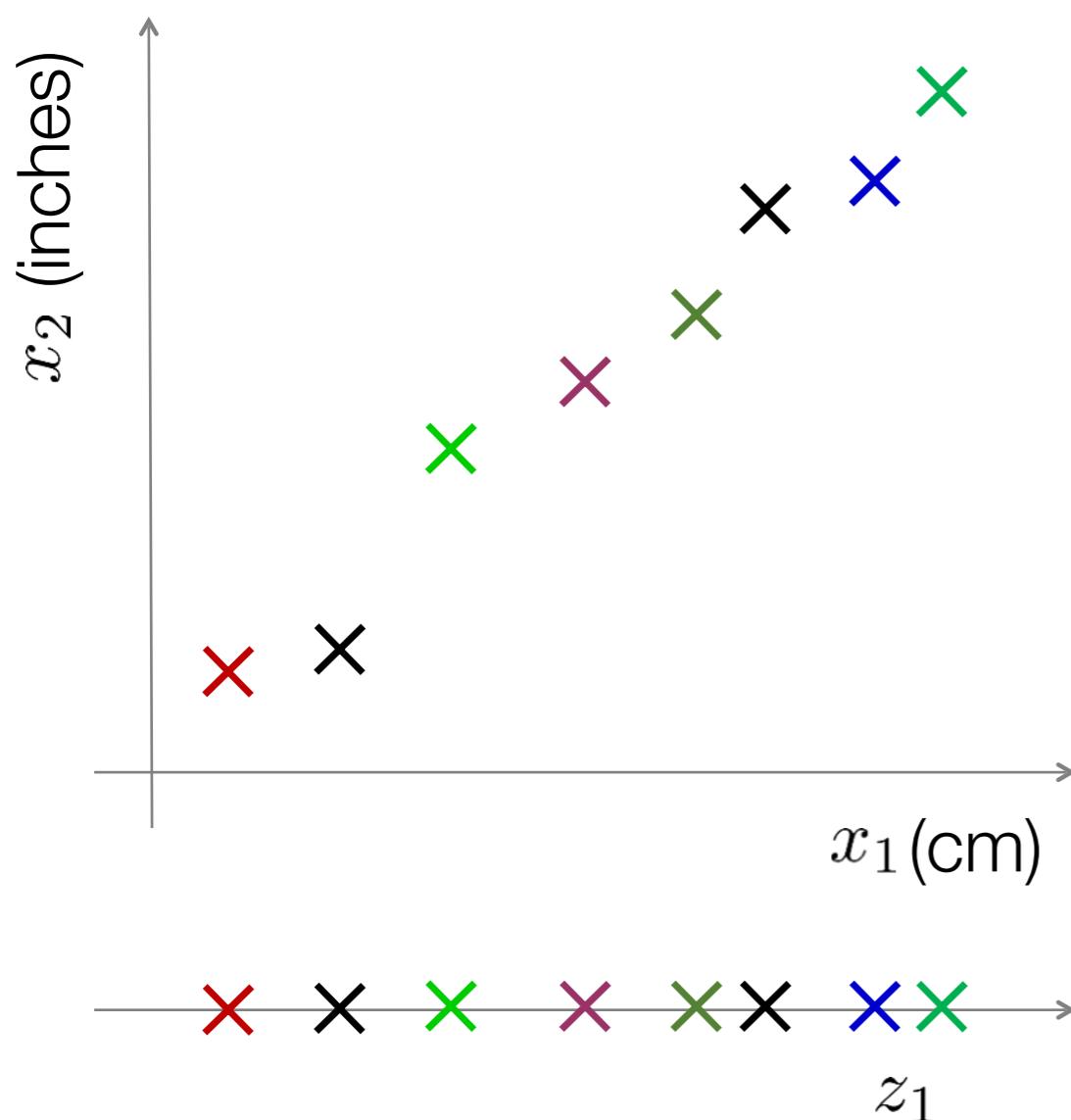
Motivation I: Data Visualization

- Is there a representation better than the coordinate axes?
- Is it really necessary to show all the 53 dimensions?
 - ... what if there are strong correlations between the features?
- How could we find the **smallest** subspace of the 53-D space that keeps the **most information** about the original data?

Motivation II: Data Compression



Motivation II: Data Compression

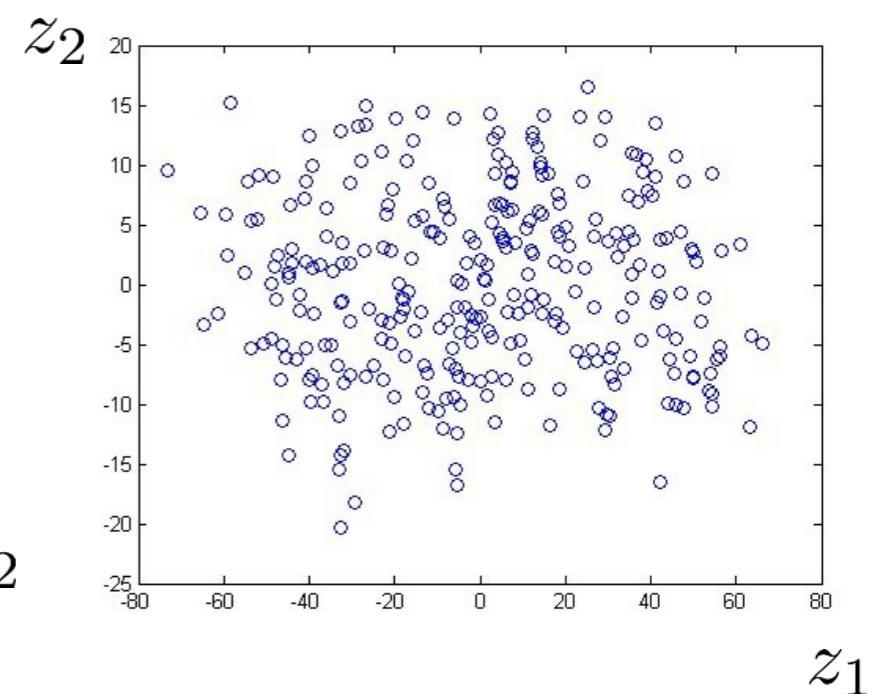
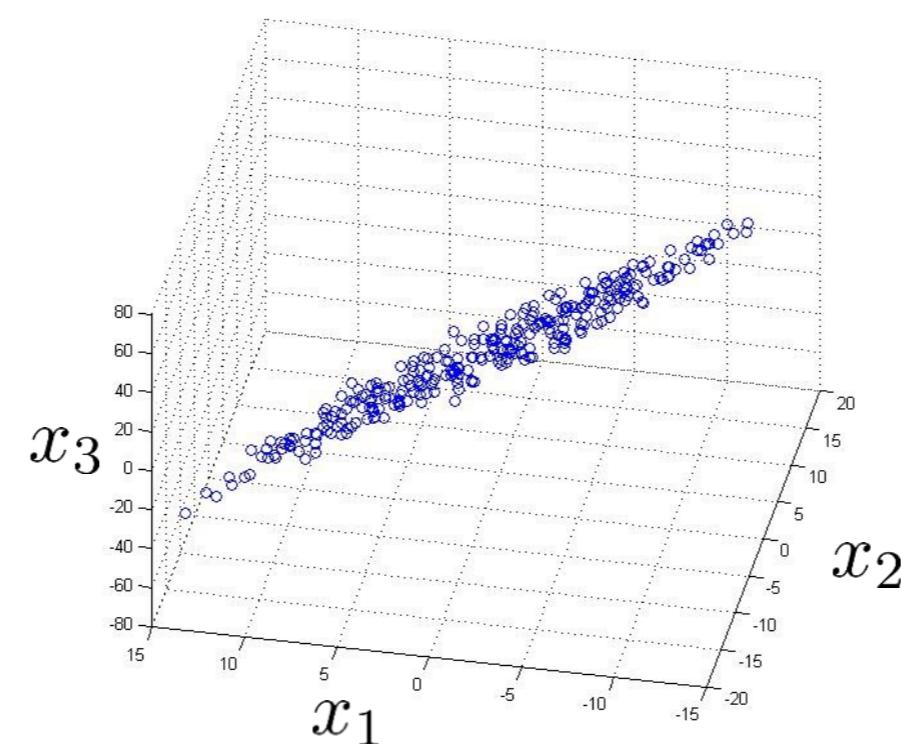
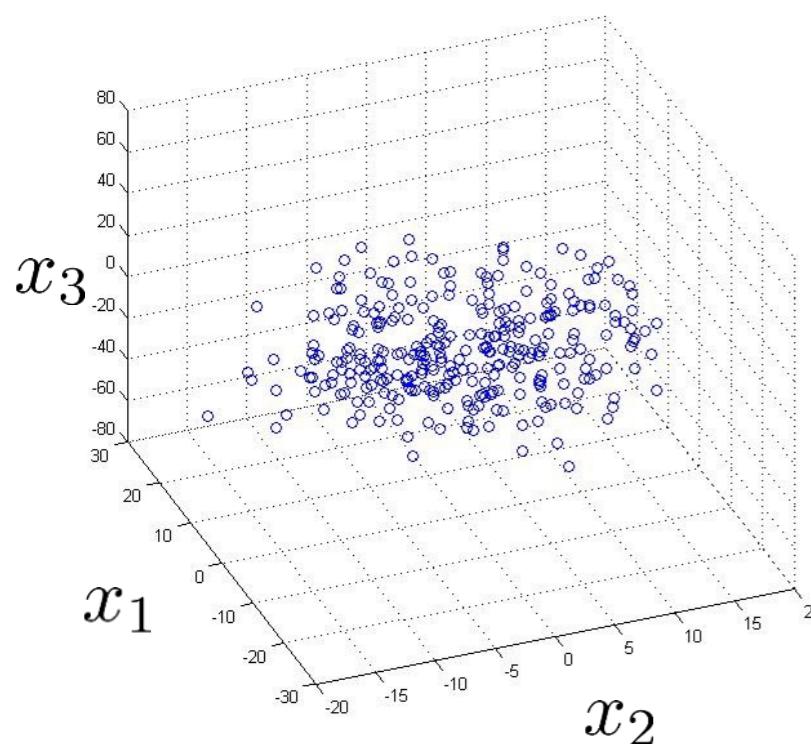


Reduce data from 2D to 1D

$$\begin{array}{ll} x^{(1)} & \rightarrow z^{(1)} \\ x^{(2)} & \rightarrow z^{(2)} \\ \vdots & \\ x^{(m)} & \rightarrow z^{(m)} \end{array}$$

Motivation II: Data Compression

Reduce data from 3D to 2D



Dimensionality Reduction

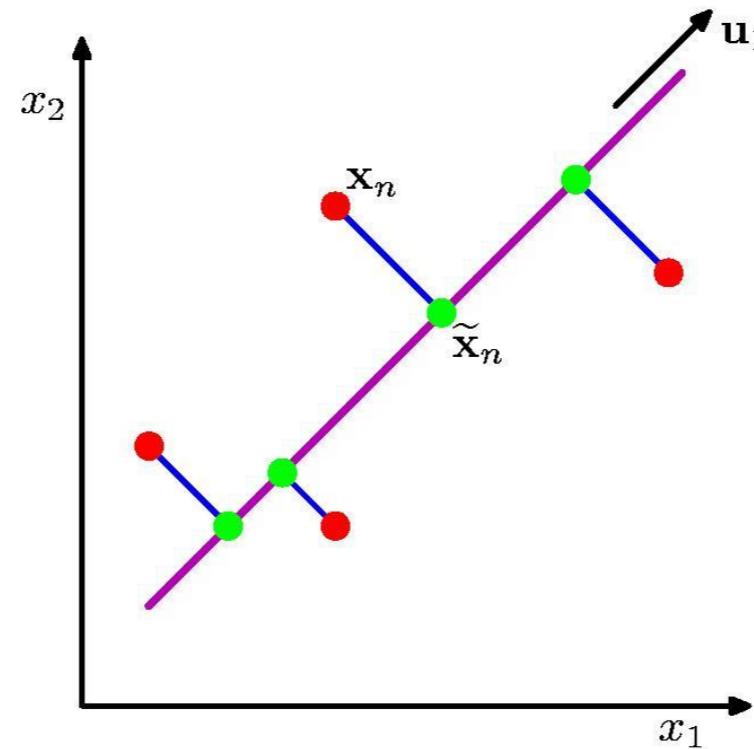
- Clustering
 - One way to summarize a complex real-valued data point with a single categorical variable
- Dimensionality reduction
 - Another way to simplify complex high-dimensional data
 - Summarize data with a lower dimensional real valued vector

- 
- Given data points in d dimensions
 - Convert them to data points in $r < d$ dims
 - With minimal loss of information

Principal Component Analysis

Principal Component Analysis

PCA:



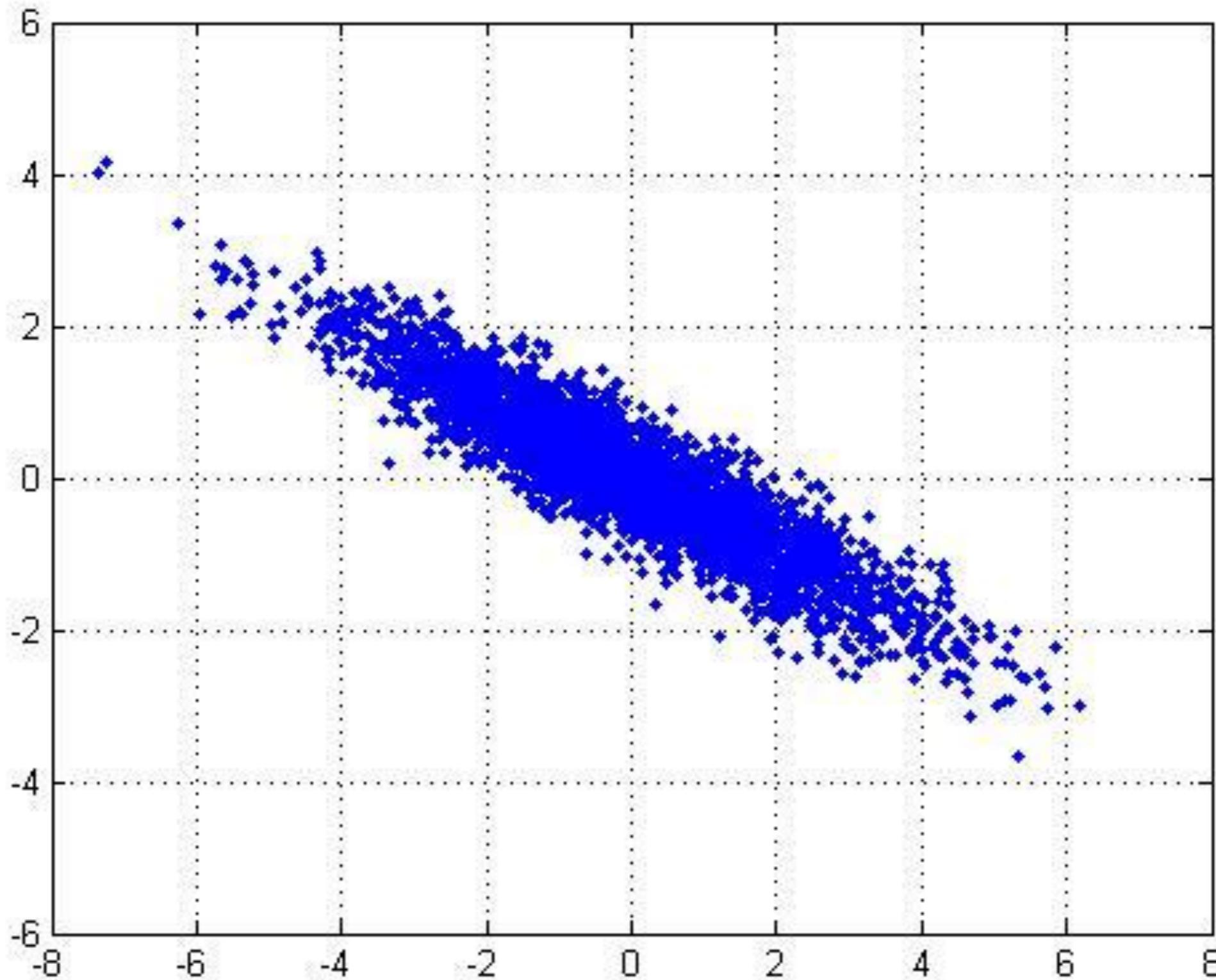
Orthogonal projection of the data onto a lower-dimension linear space that...

- maximizes variance of projected data (**purple** line)
- minimizes mean squared distance between
 - data point and
 - projections (sum of **blue** lines)

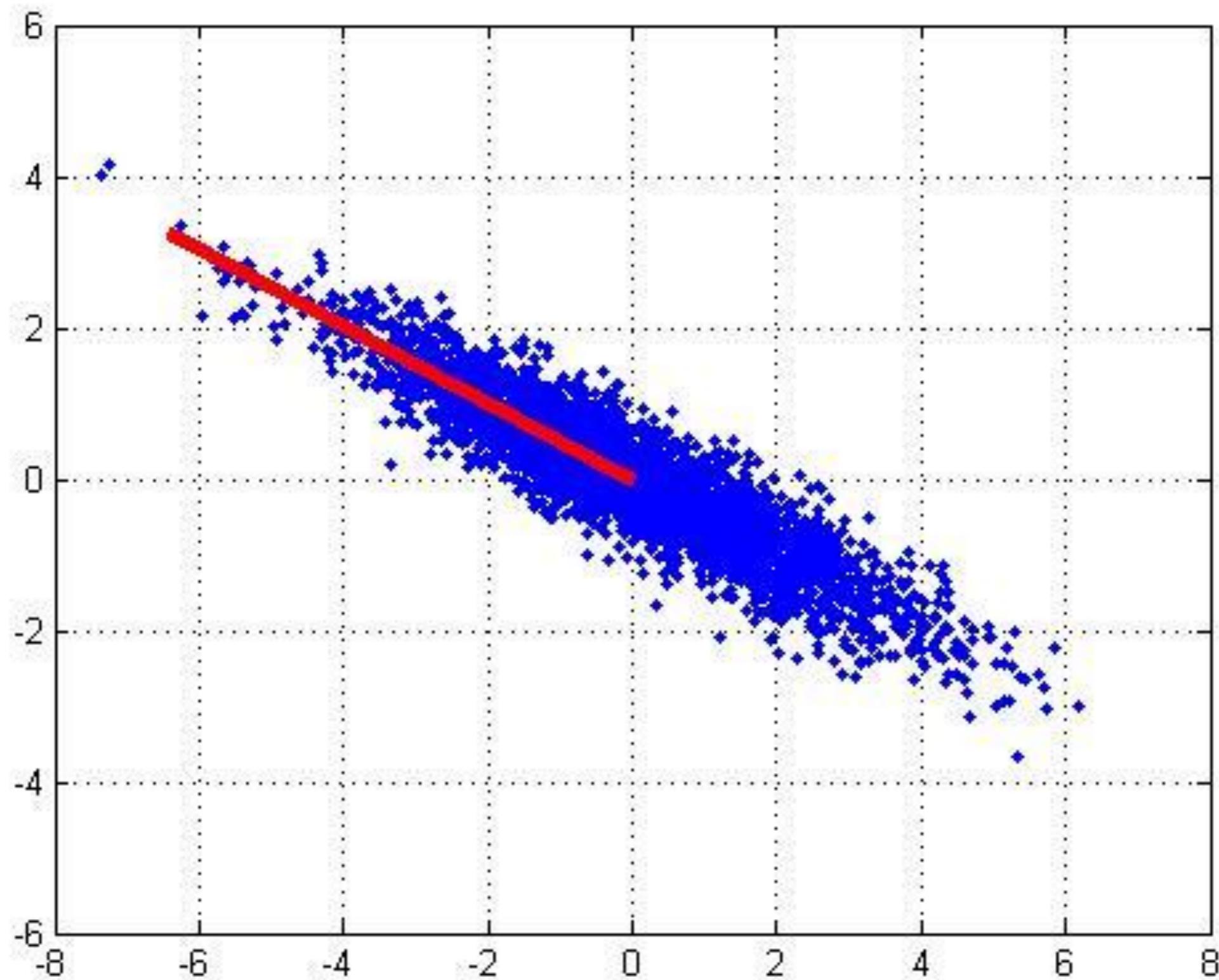
Principal Component Analysis

- PCA Vectors originate from the center of mass.
- Principal component #1: points in the direction of the **largest variance**.
- Each subsequent principal component
 - is **orthogonal** to the previous ones, and
 - points in the directions of the **largest variance of the residual subspace**

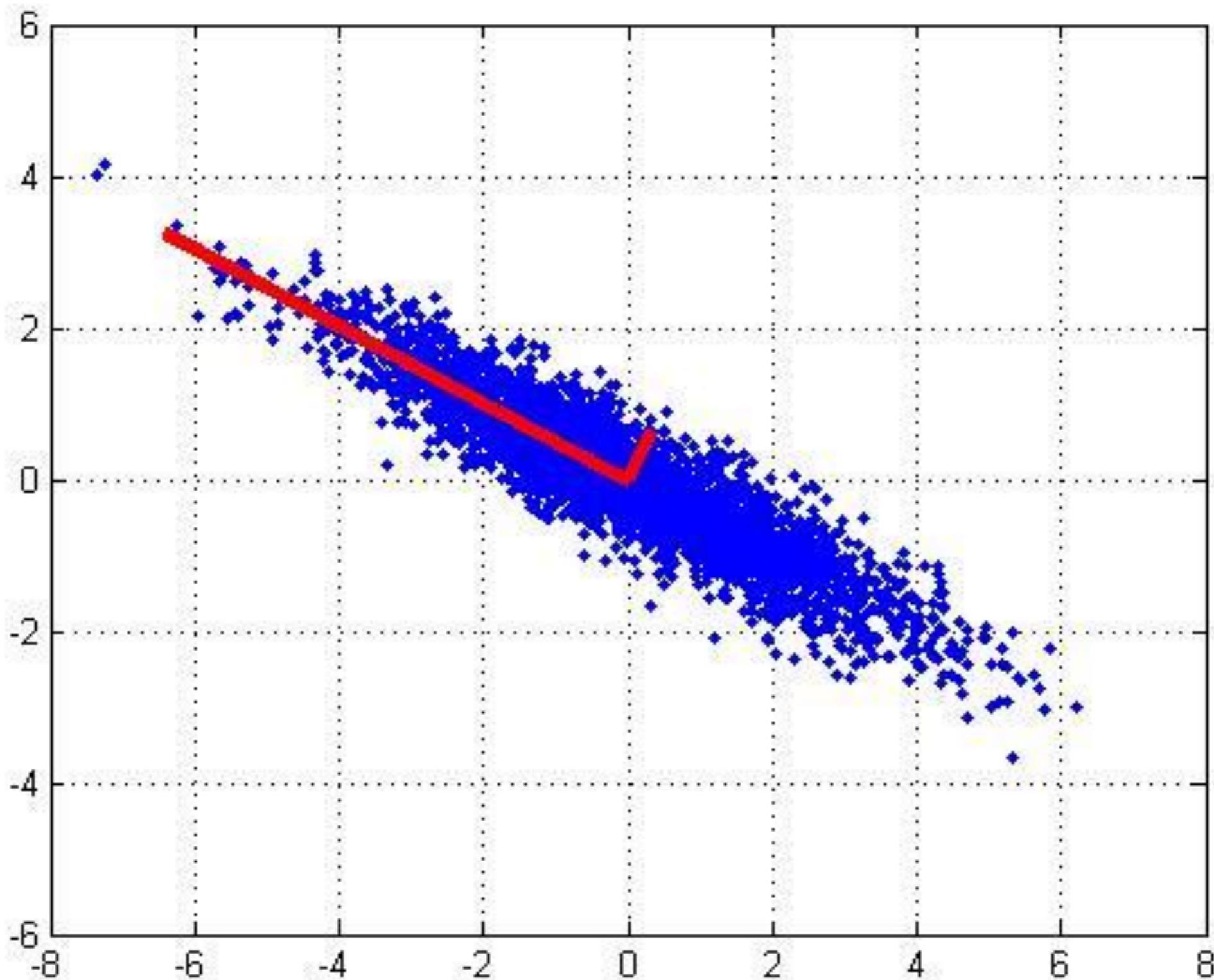
2D Gaussian dataset



1st PCA axis



2nd PCA axis



PCA algorithm I (sequential)

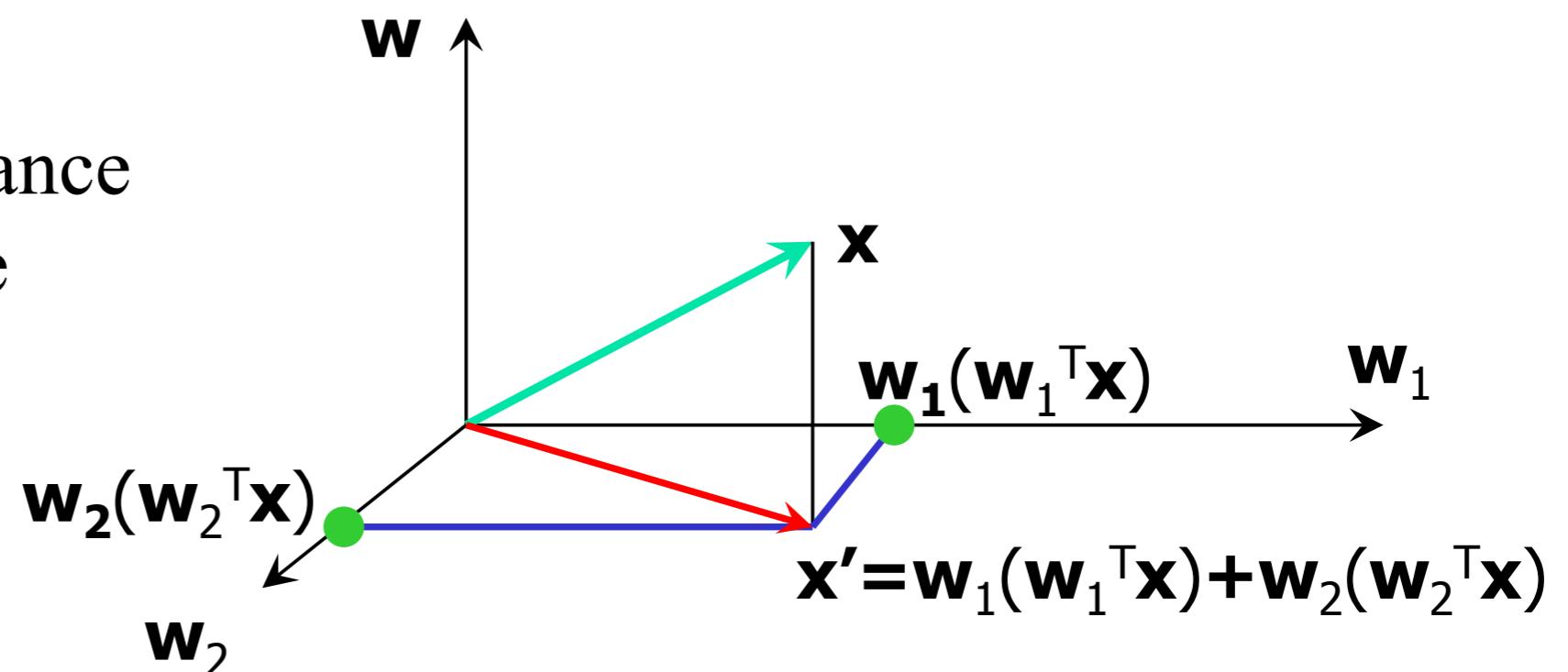
Given the **centered** data $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$, compute the principal vectors:

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

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

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

We maximize the variance
of the projection in the
residual subspace



PCA algorithm II

(sample covariance matrix)

- Given data $\{\mathbf{x}_1, \dots, \mathbf{x}_m\}$, compute covariance matrix Σ

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

where

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

- PCA basis vectors** = the eigenvectors of Σ
- Larger eigenvalue \Rightarrow more important eigenvectors

Reminder: Eigenvector and Eigenvalue

$$Ax = \lambda x$$

A: Square matrix

λ : Eigenvector or characteristic vector

x : Eigenvalue or characteristic value

Show $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$ is an eigenvector for $A = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix}$

Solution : $Ax = \begin{bmatrix} 2 & -4 \\ 3 & -6 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

But for $\lambda = 0$, $\lambda x = 0 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

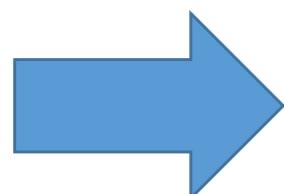
Thus, x is an eigenvector of A , and $\lambda = 0$ is an eigenvalue.

Reminder: Eigenvector and Eigenvalue

$$\mathbf{Ax} = \lambda\mathbf{x} \quad \rightarrow \quad \begin{aligned}\mathbf{Ax} - \lambda\mathbf{x} &= \mathbf{0} \\ (\mathbf{A} - \lambda\mathbf{I})\mathbf{x} &= \mathbf{0}\end{aligned}$$

If we define a new matrix \mathbf{B} : \rightarrow
$$\begin{aligned}\mathbf{B} &= \mathbf{A} - \lambda\mathbf{I} \\ \mathbf{Bx} &= \mathbf{0}\end{aligned}$$

If \mathbf{B} has an inverse: \rightarrow
$$\mathbf{x} = \mathbf{B}^{-1}\mathbf{0} = \mathbf{0} \times \text{BUT! an eigenvector cannot be zero!!}$$



\mathbf{x} will be an eigenvector of \mathbf{A} if and only if \mathbf{B} does not have an inverse, or equivalently $\det(\mathbf{B})=0$:

$$\boxed{\det(\mathbf{A} - \lambda\mathbf{I}) = 0}$$

Reminder: Eigenvector and Eigenvalue

Example 1: Find the eigenvalues of

$$A = \begin{bmatrix} 2 & -12 \\ 1 & -5 \end{bmatrix}$$

$$\begin{aligned} |\lambda I - A| &= \begin{vmatrix} \lambda - 2 & 12 \\ -1 & \lambda + 5 \end{vmatrix} = (\lambda - 2)(\lambda + 5) + 12 \\ &= \lambda^2 + 3\lambda + 2 = (\lambda + 1)(\lambda + 2) \end{aligned}$$

two eigenvalues: $-1, -2$

Note: The roots of the characteristic equation can be repeated. That is, $\lambda_1 = \lambda_2 = \dots = \lambda_k$. If that happens, the eigenvalue is said to be of multiplicity k .

Example 2: Find the eigenvalues of

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$|\lambda I - A| = \begin{vmatrix} \lambda - 2 & -1 & 0 \\ 0 & \lambda - 2 & 0 \\ 0 & 0 & \lambda - 2 \end{vmatrix} = (\lambda - 2)^3 = 0$$

$\lambda = 2$ is an eigenvector of multiplicity 3.

PCA algorithm II (sample covariance matrix)

Goal: Find r -dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = A(x - \mu)$$

where y is the new point, x is the old one,
and the rows of A are the eigenvectors

PCA algorithm III (SVD of the data matrix)

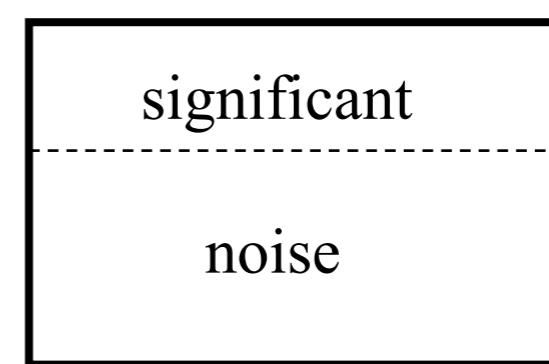
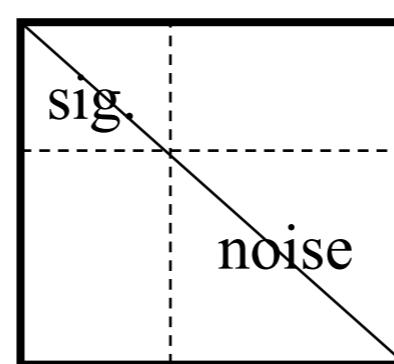
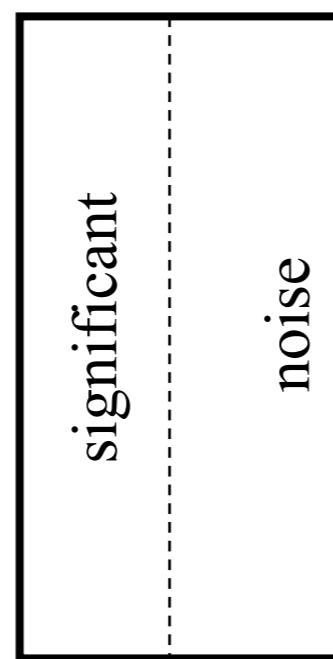
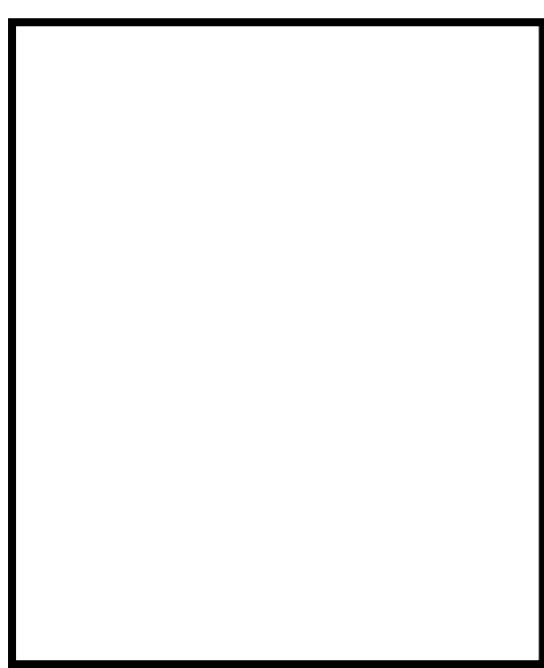
Singular Value Decomposition of the **centered** data matrix \mathbf{X} .

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

m : number of instances,
 N : dimension

$$\mathbf{X}_{\text{features} \times \text{samples}} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$

$$\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$



PCA algorithm III

- **Columns of \mathbf{U}**
 - the principal vectors, $\{ \mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)} \}$
 - orthogonal and has unit norm – so $\mathbf{U}^T \mathbf{U} = \mathbf{I}$
 - Can reconstruct the data using linear combinations of $\{ \mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)} \}$
- **Matrix \mathbf{S}**
 - Diagonal
 - Shows importance of each eigenvector
- **Columns of \mathbf{V}^T**
 - The coefficients for reconstructing the samples

Applications

Face Recognition

Face Recognition

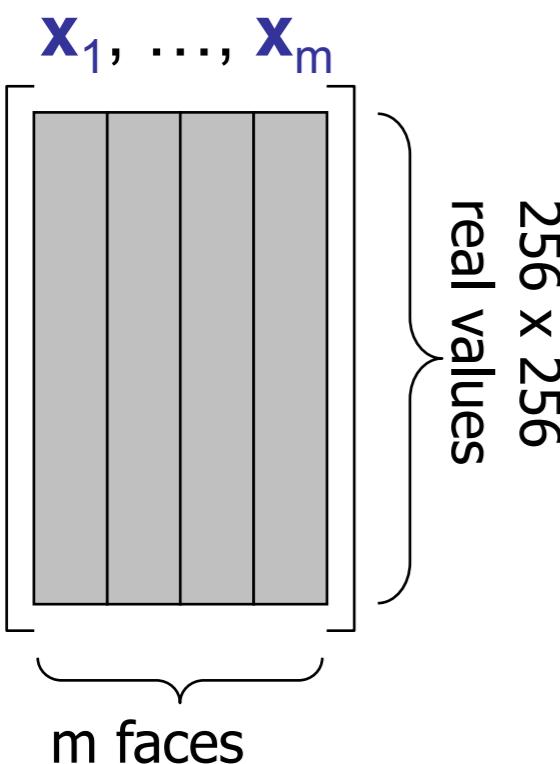
- Want to identify specific person, based on facial image
- Robust to glasses, lighting, ...
 - Can't just use the given 256×256 pixels



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 $\mathbb{R}^{256 \times 256}$ (view as 64K dim vector)
- Form $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_m]$ **centered** data mtx
- Compute $\Sigma = \mathbf{X}\mathbf{X}^T$
- Problem: Σ is $64K \times 64K \dots$ **HUGE!!!**

A Clever Workaround

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

Proof:

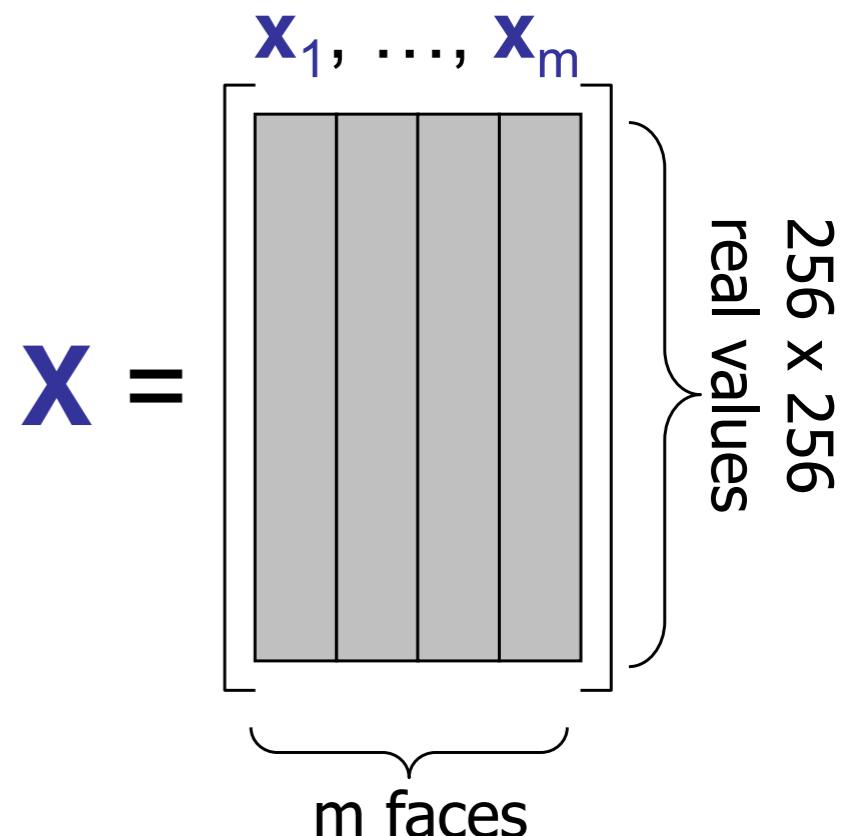
$$\mathbf{L} \mathbf{v} = \gamma \mathbf{v}$$

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

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

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

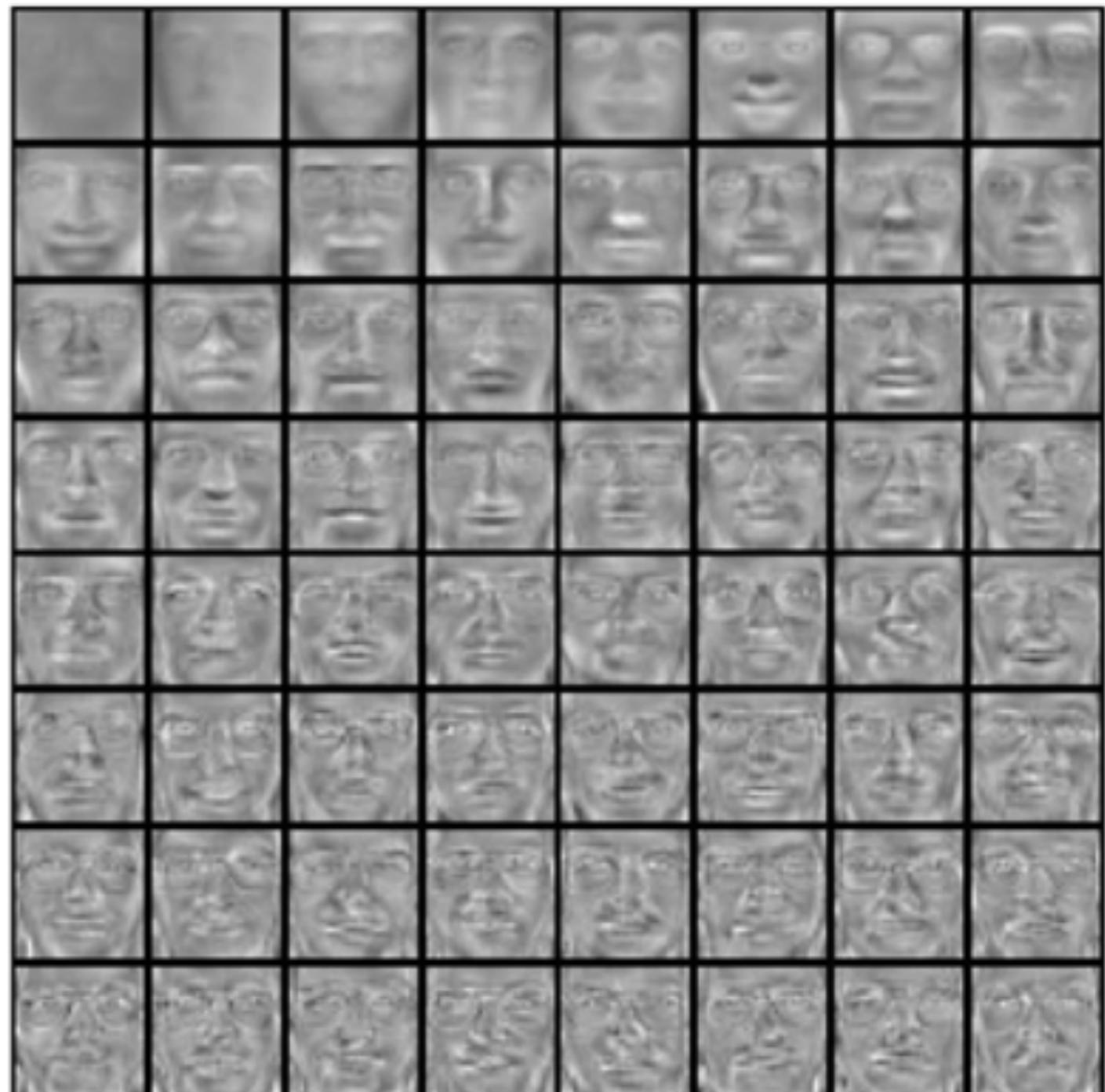
$$\Sigma(\mathbf{Xv}) = \gamma(\mathbf{Xv})$$



Eigenfaces Example

Top eigenvectors: u_1, \dots, u_k

Mean: μ



Representation and Reconstruction

- Face \mathbf{x} in “face space” coordinates:



$$\begin{aligned}\mathbf{x} &\rightarrow [\mathbf{u}_1^T(\mathbf{x} - \mu), \dots, \mathbf{u}_k^T(\mathbf{x} - \mu)] \\ &= w_1, \dots, w_k\end{aligned}$$

- Reconstruction:

$$\begin{aligned}\hat{\mathbf{x}} &= \mathbf{\mu} + \begin{matrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \\ \vdots \end{matrix} w_1 + w_2 + w_3 + w_4 + \dots\end{aligned}$$

The reconstruction equation shows the original face $\hat{\mathbf{x}}$ being approximated by the mean face $\mathbf{\mu}$ plus a sum of weighted principal components \mathbf{u}_i (represented as vertical strips) and their corresponding weights w_i .

Principle Components (Method B)



Principle Components (Method B)



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

When projecting strange data

- Original images
- Reconstruction doesn't look like the original



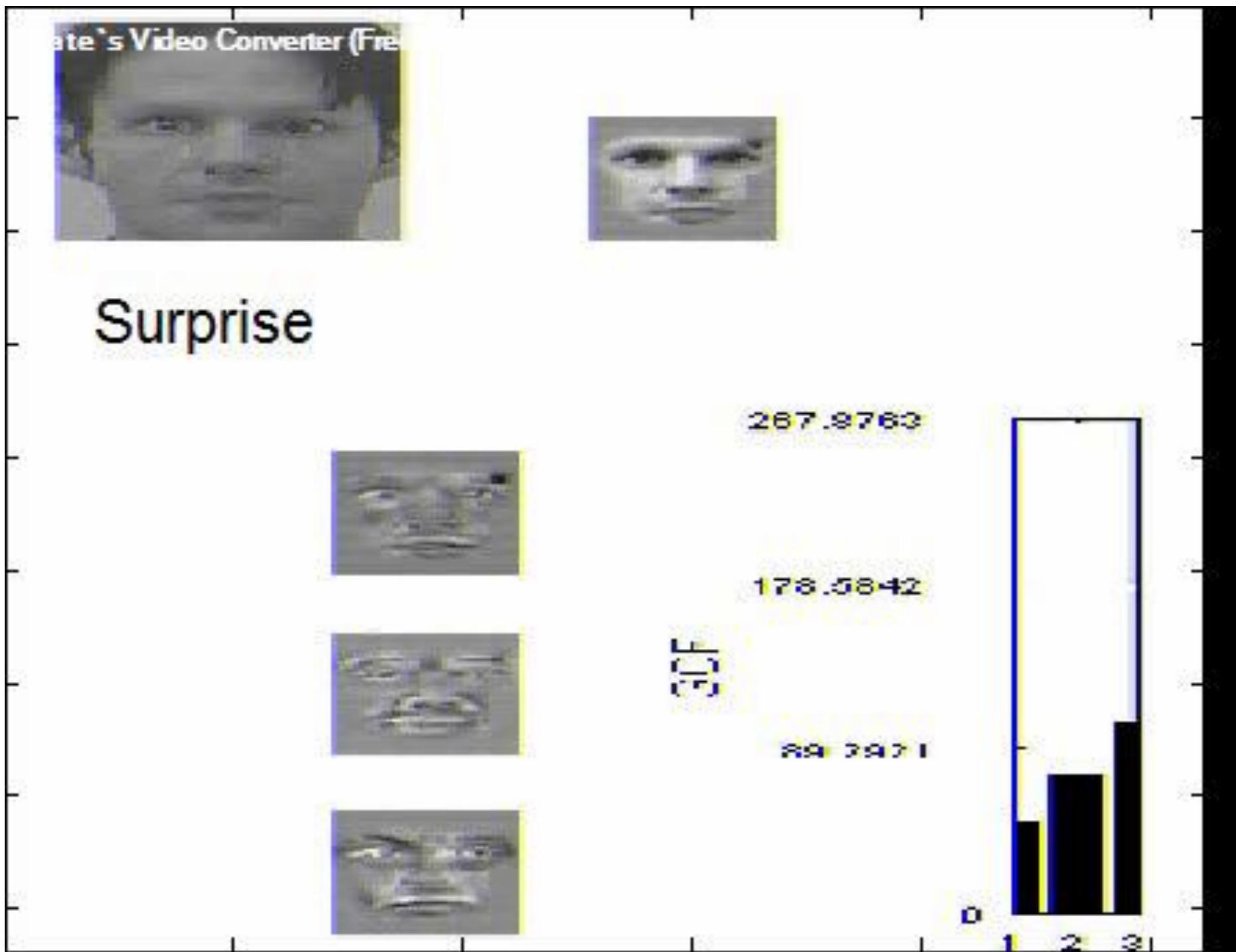
Happiness subspace (method A)



Disgust subspace (method A)



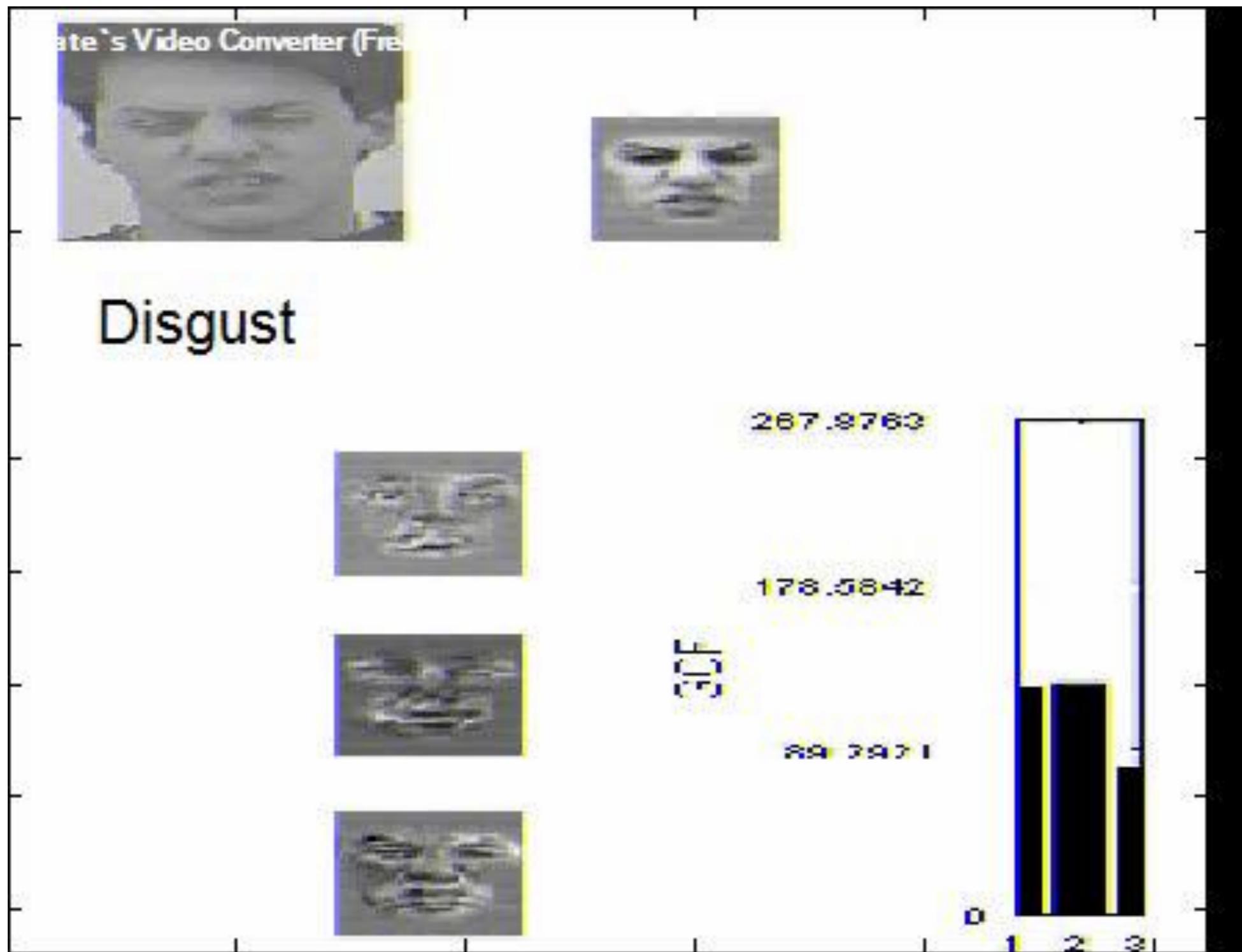
Facial Expression Recognition Movies



Facial Expression Recognition Movies



Facial Expression Recognition Movies



Shortcomings

- Requires carefully controlled data:
 - All faces centered in frame
 - Same size
 - Some sensitivity to angle
- 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

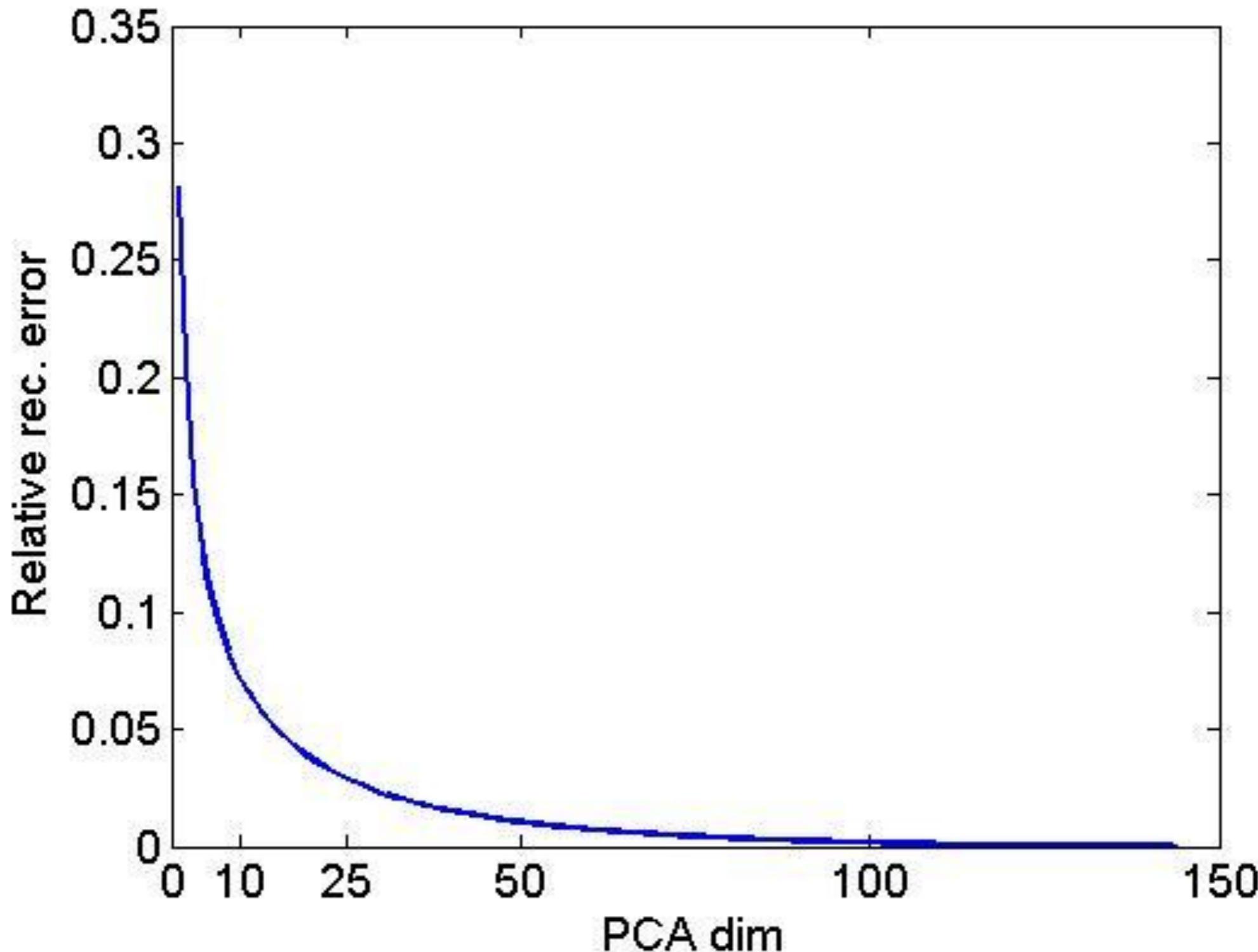
Image Compression

Original Image



- Divide the original 372x492 image into patches:
 - Each patch is an instance
- View each as a 144-D vector

L_2 error and PCA dim



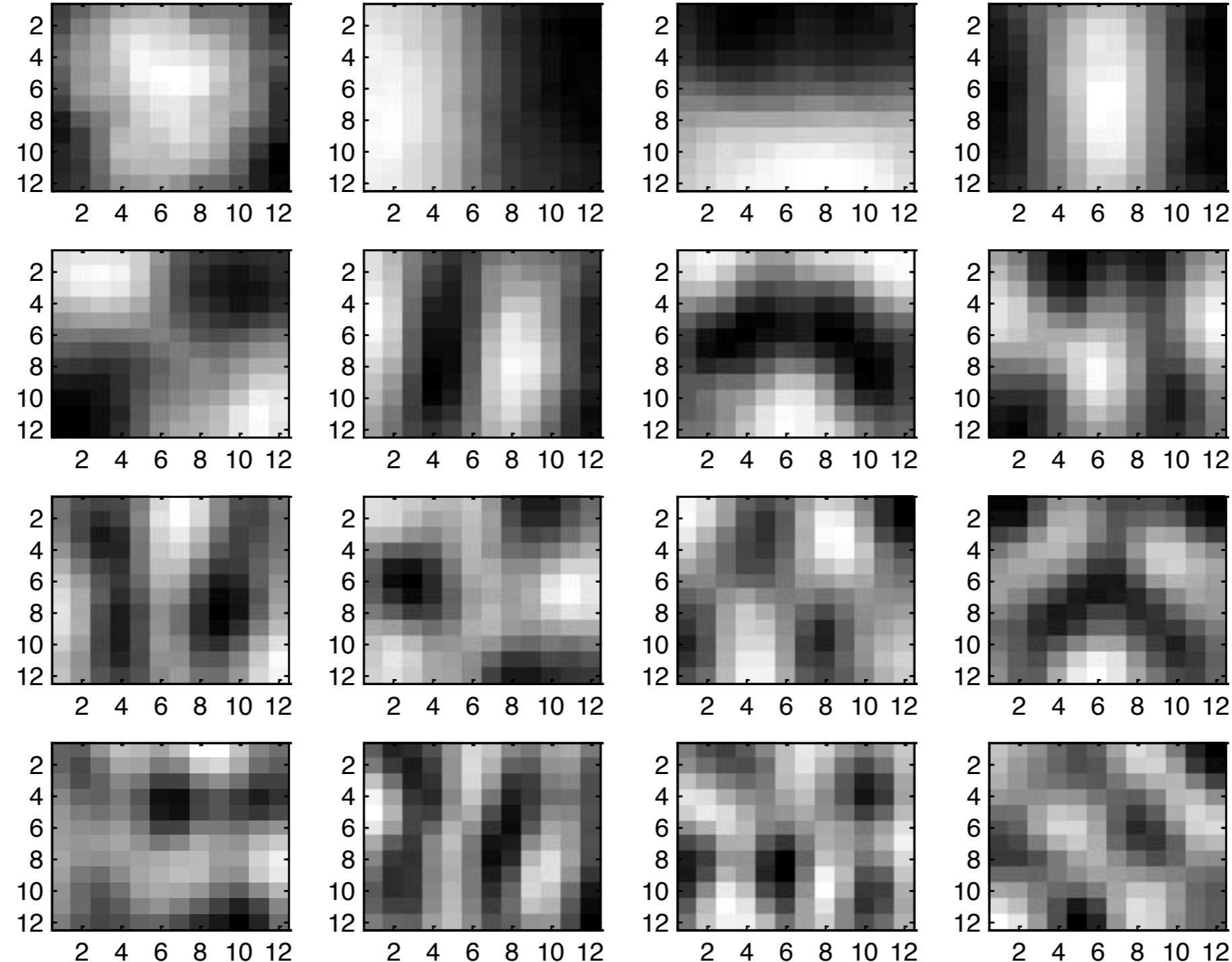
PCA compression: 144D \Rightarrow 60D



PCA compression: 144D \Rightarrow 16D



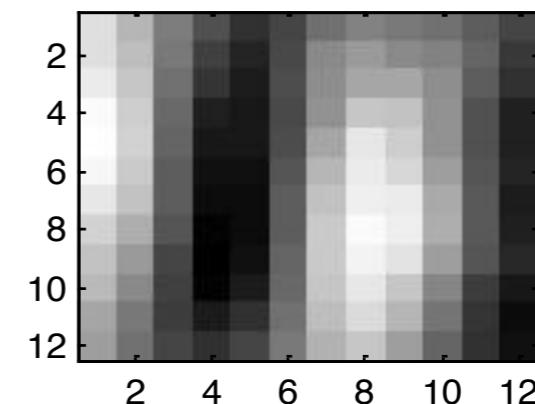
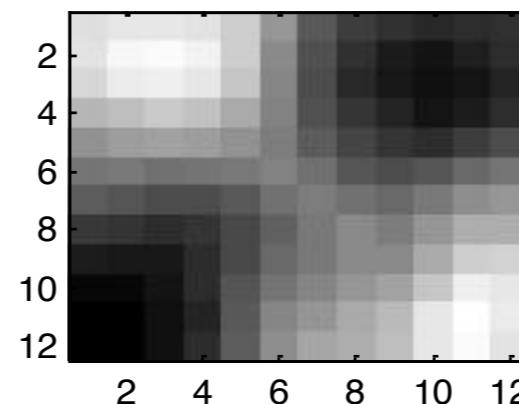
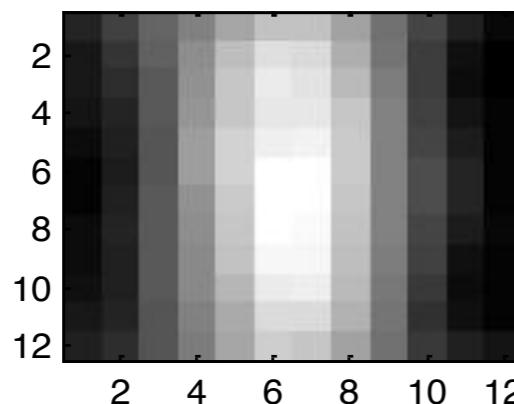
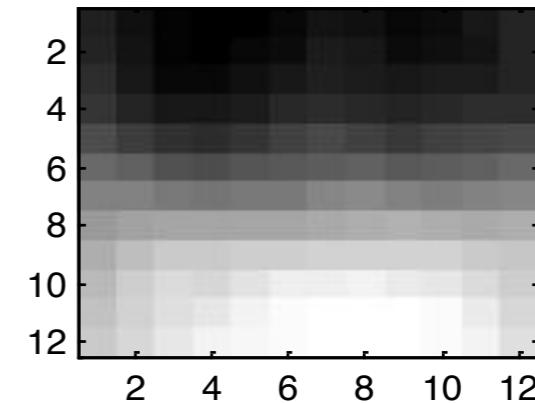
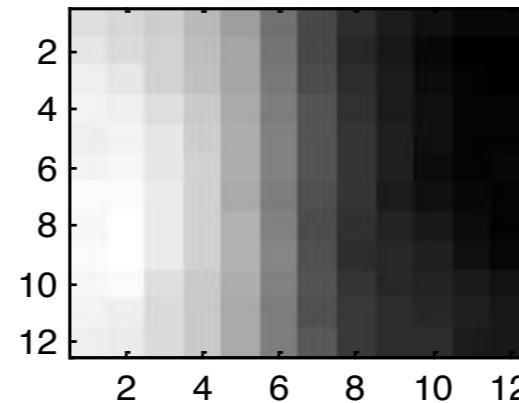
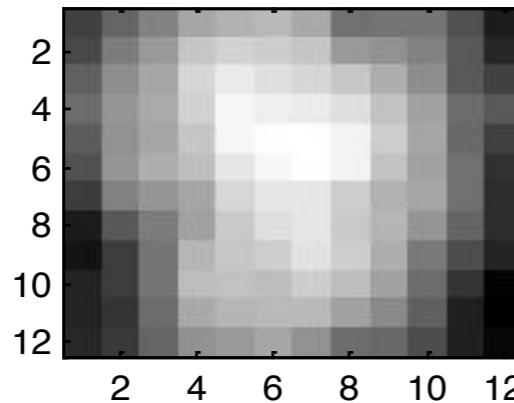
16 most important eigenvectors



PCA compression: 144D \Rightarrow 6D



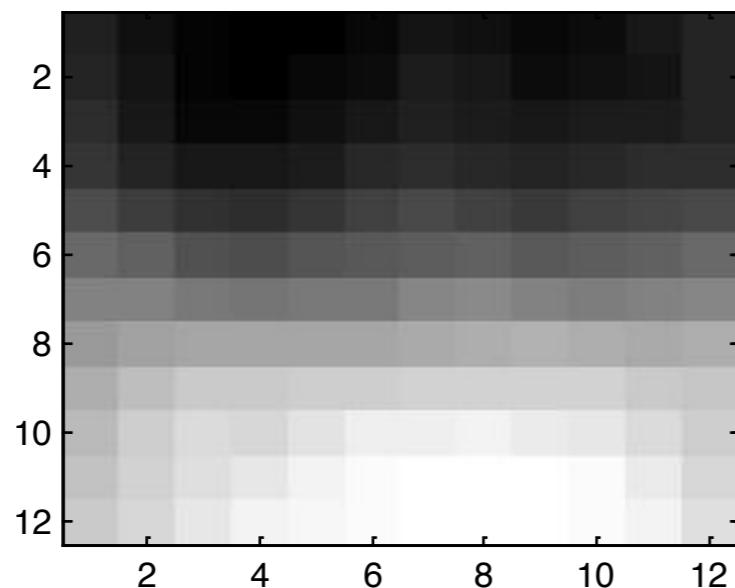
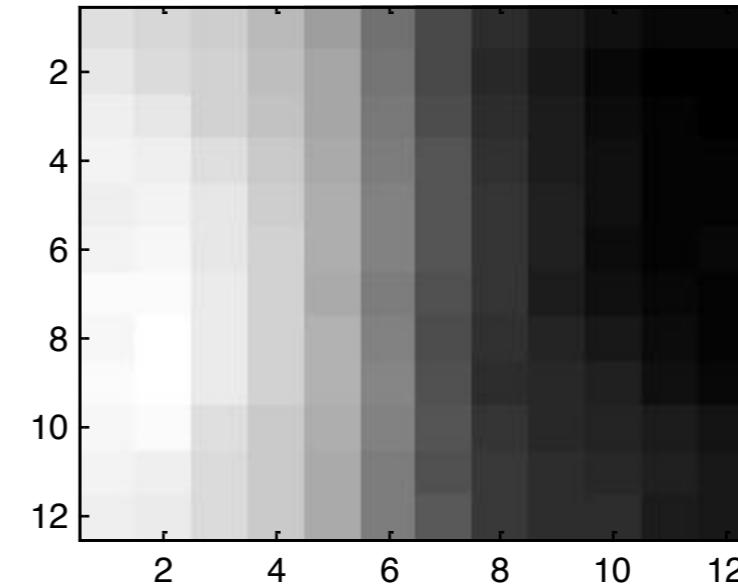
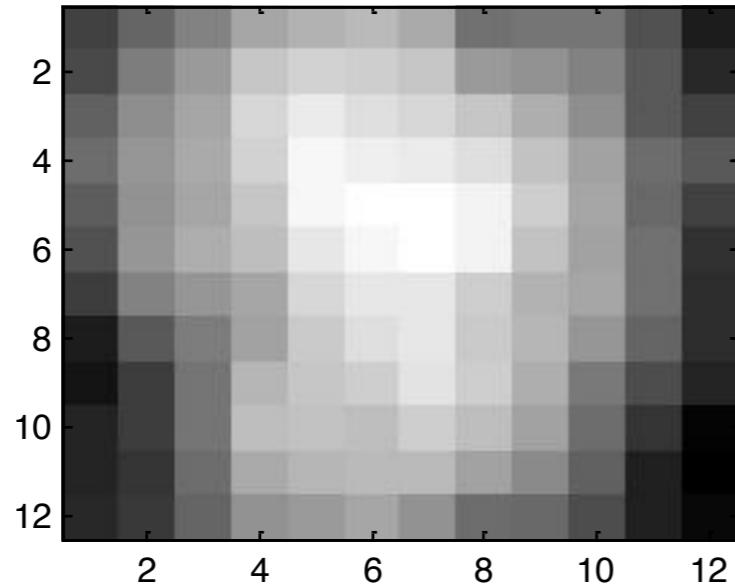
6 most important eigenvectors



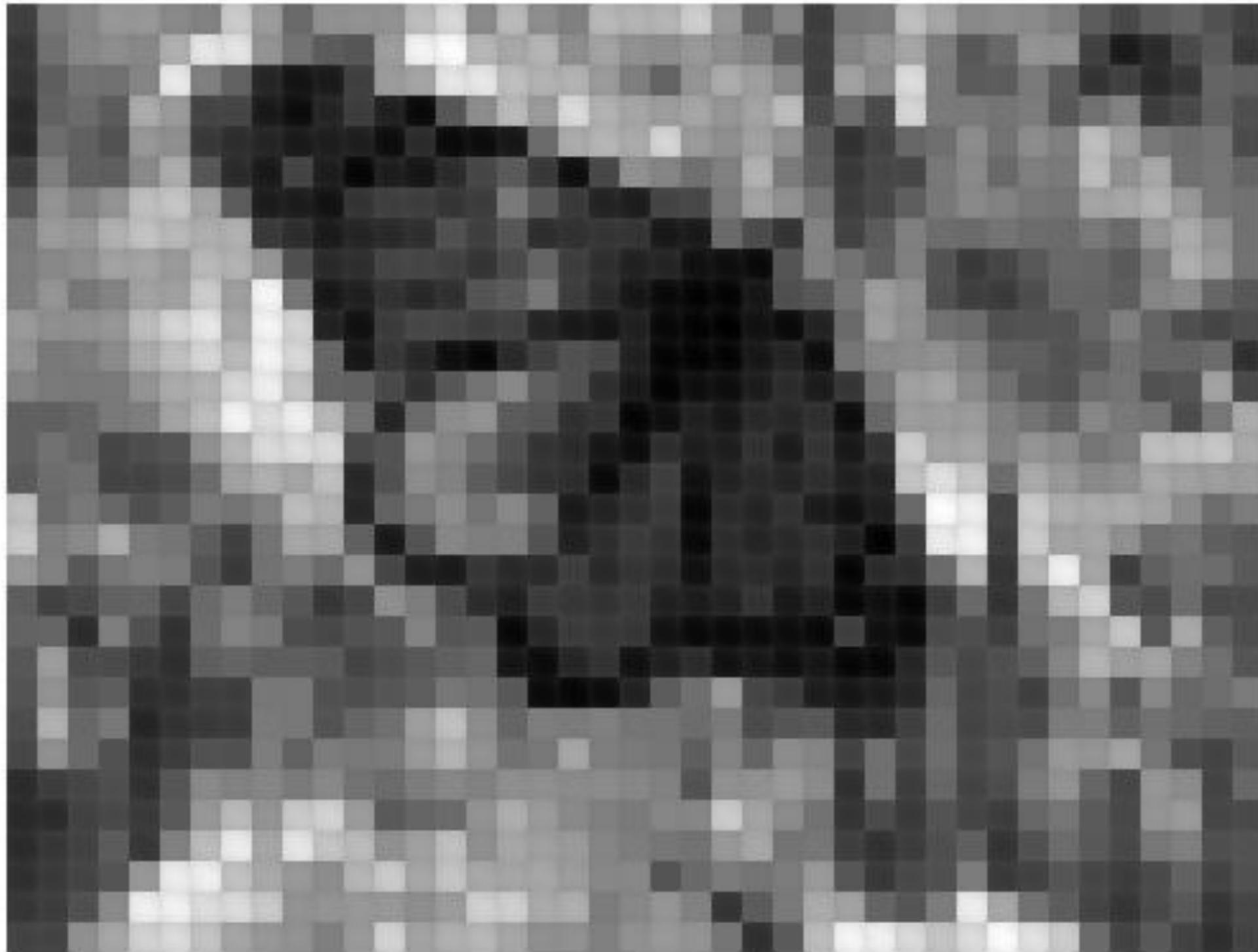
PCA compression: 144D \Rightarrow 3D



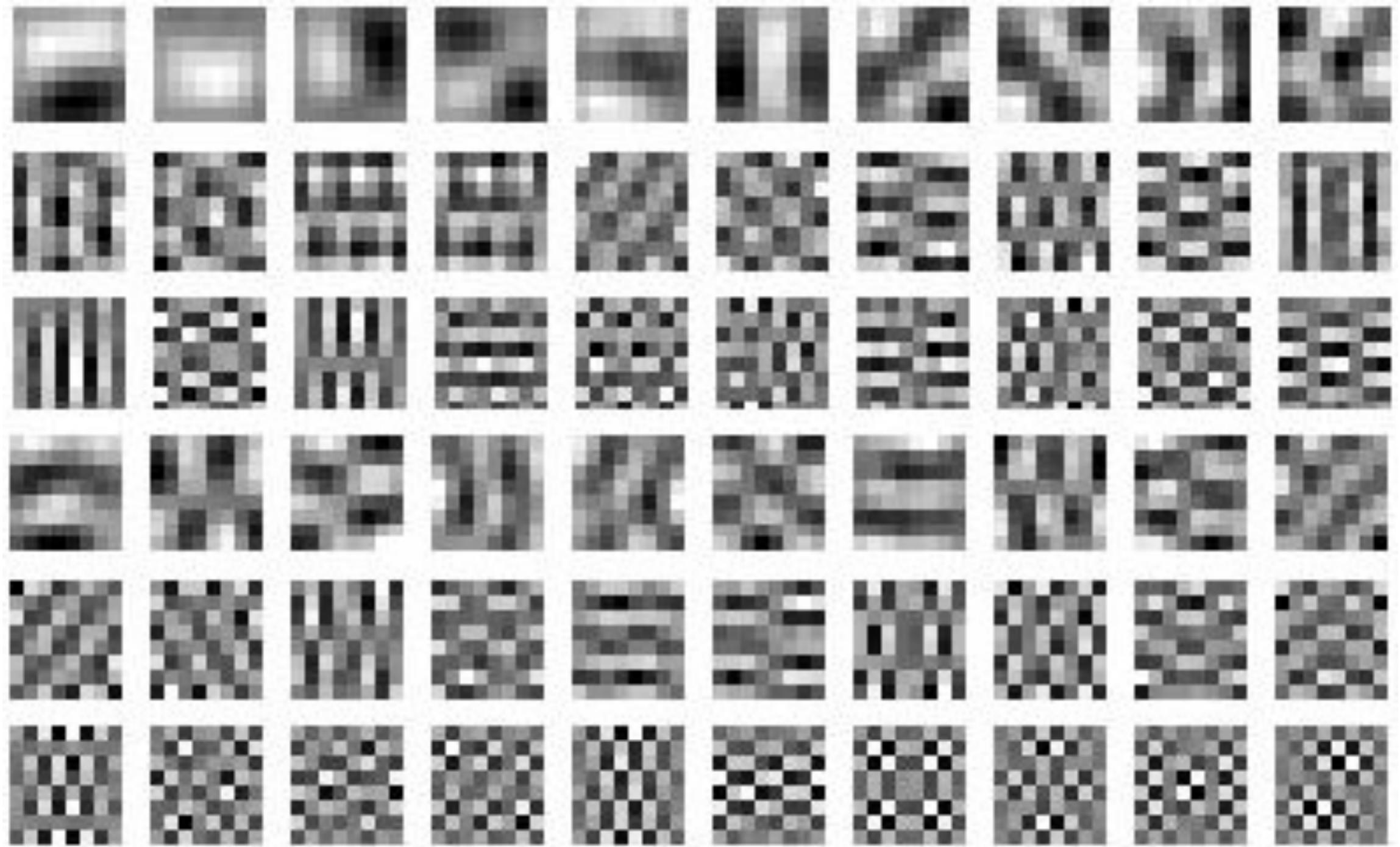
3 most important eigenvectors



PCA compression: 144D \Rightarrow 1D

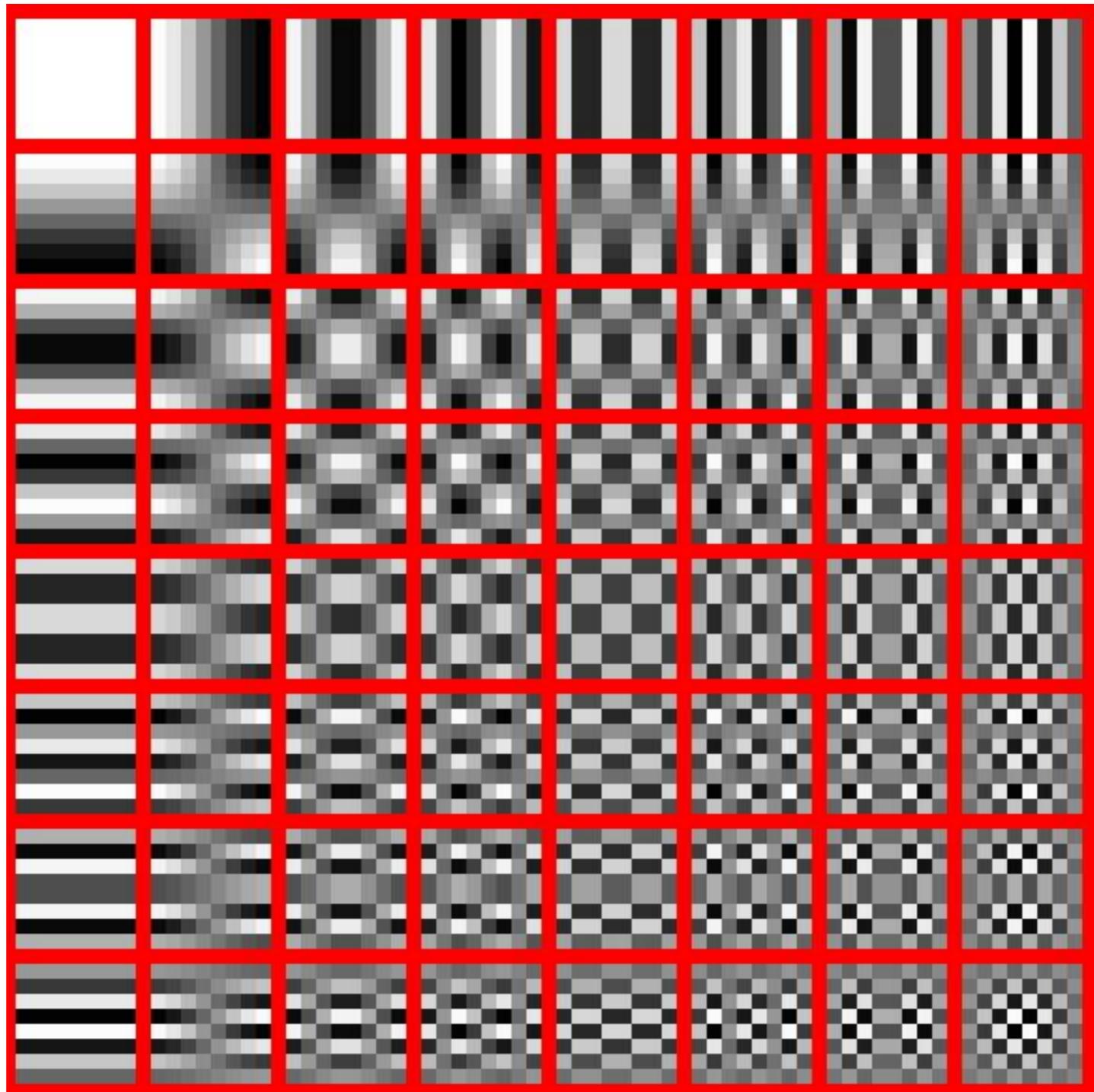


60 most important eigenvectors



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

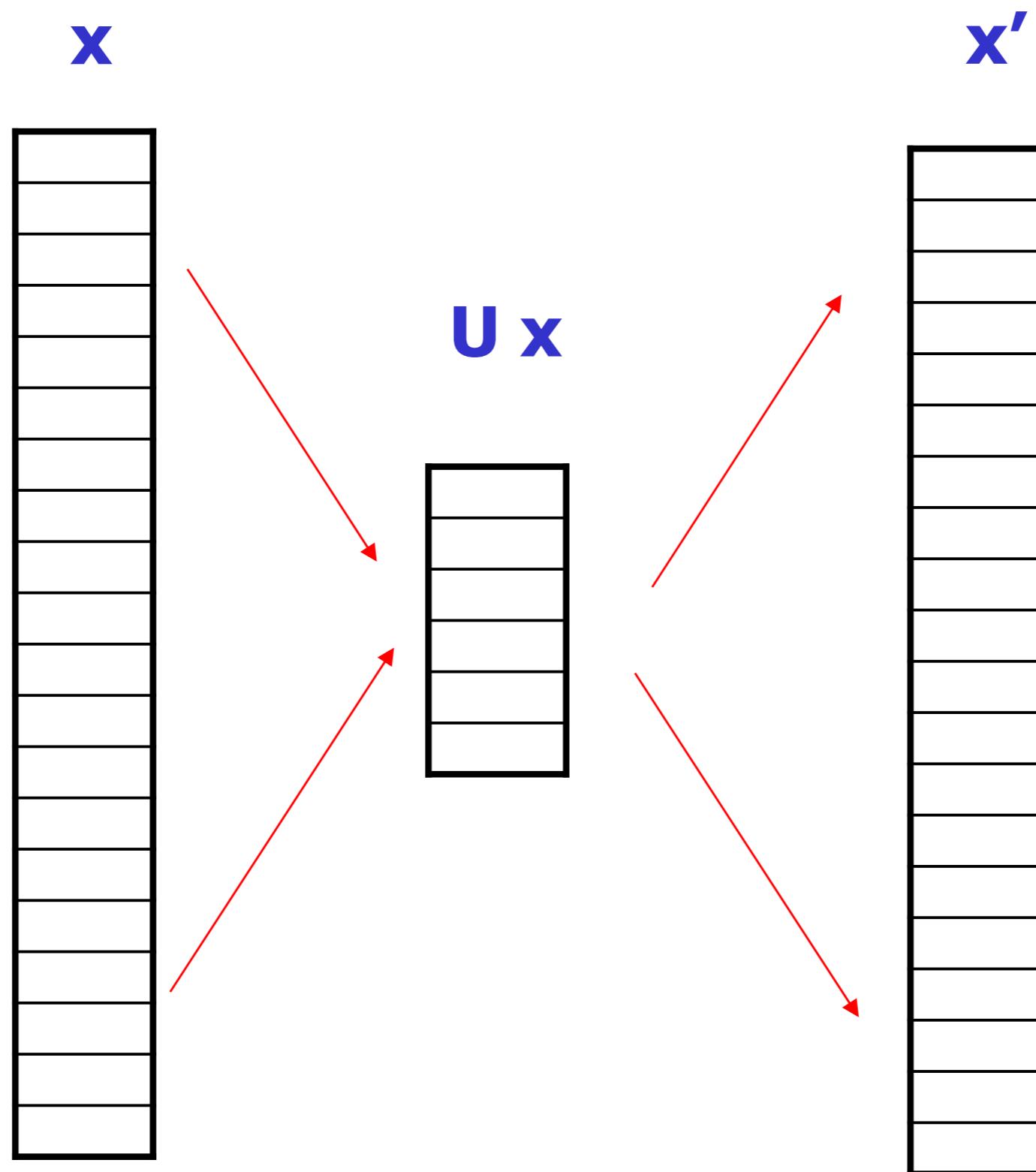
2D Discrete Cosine Basis



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

Noise Filtering

Noise Filtering



Noisy image



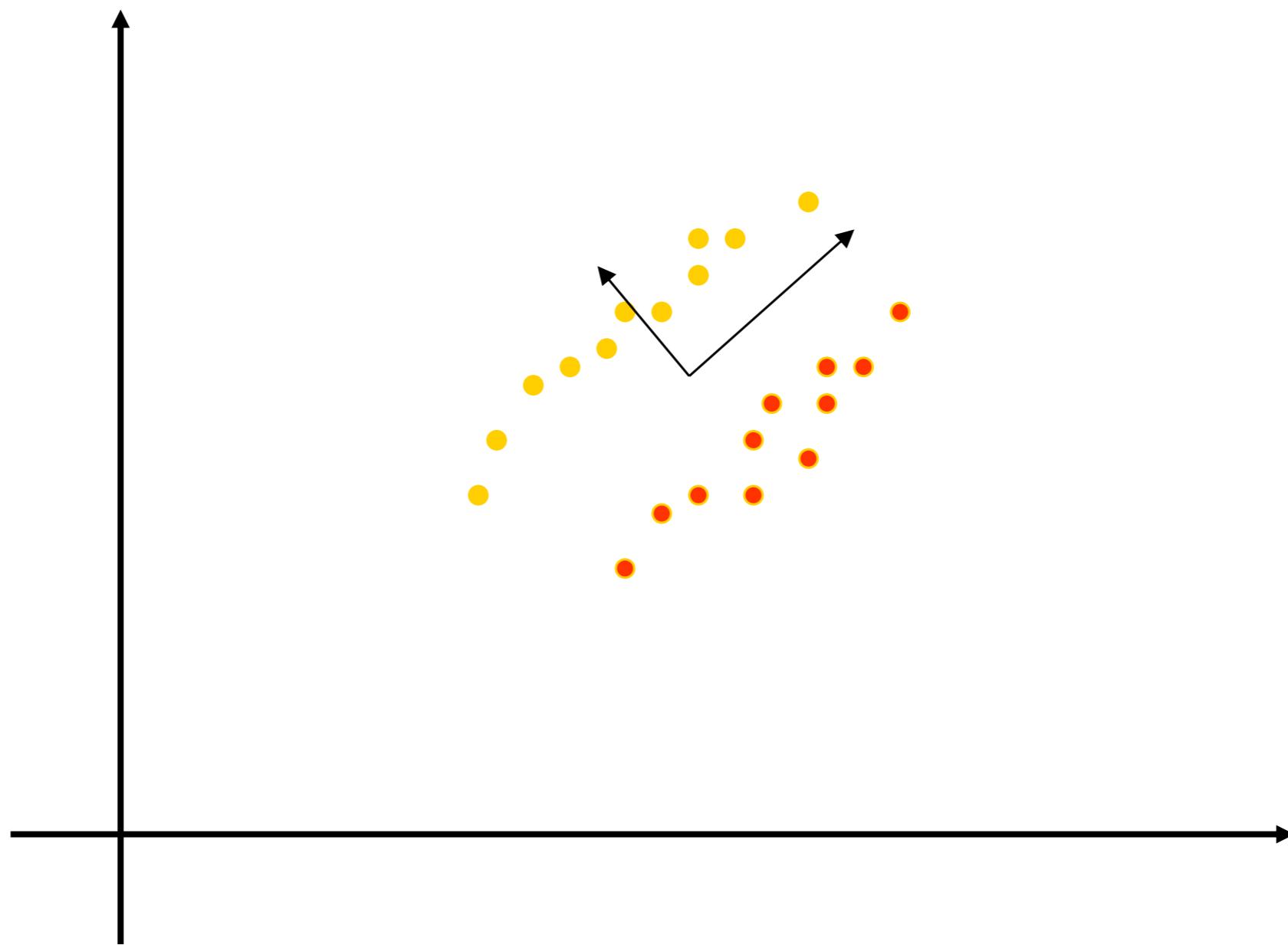
Denoised image using 15 PCA components



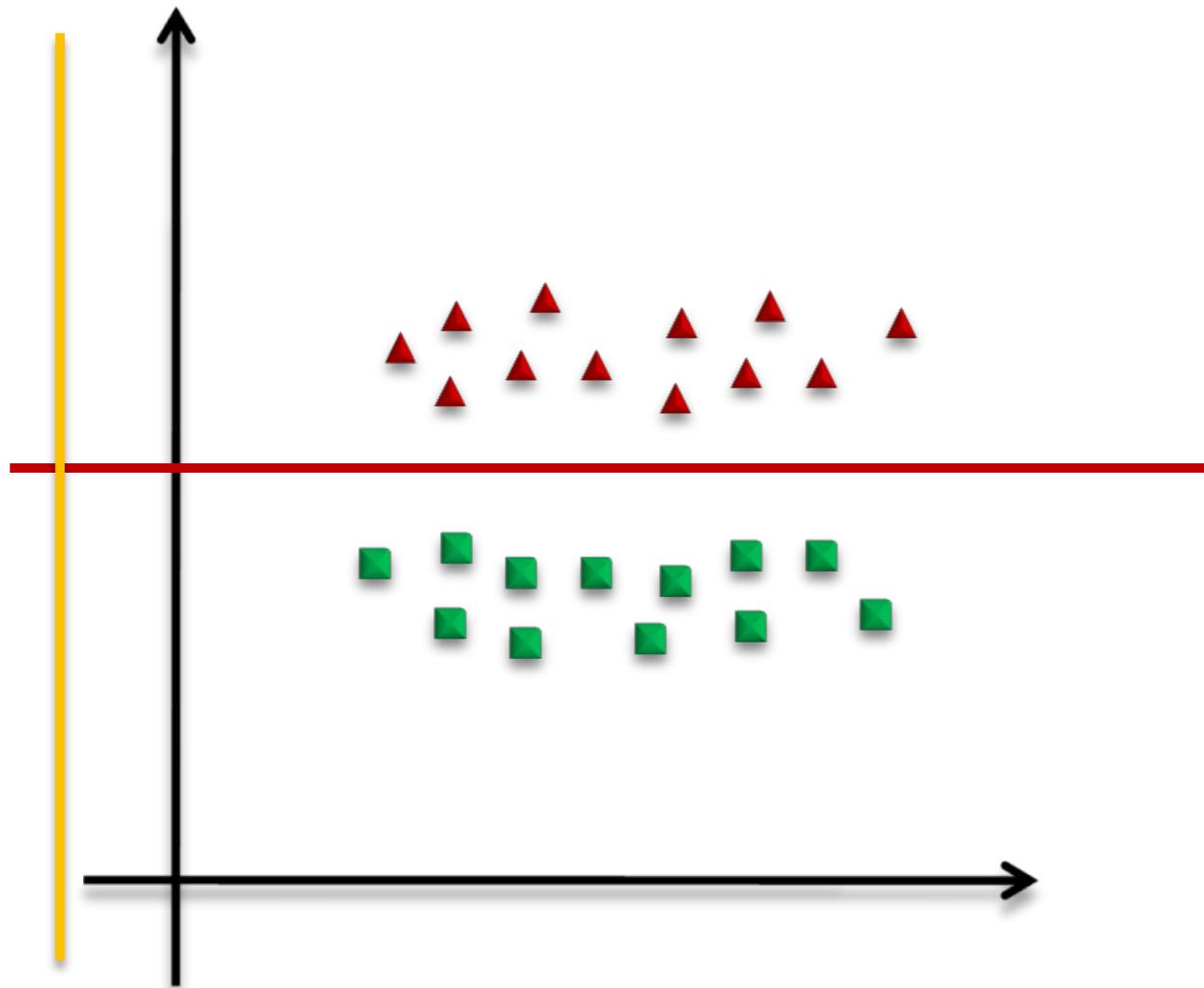
PCA Shortcomings

Problematic Data Set for PCA

- PCA doesn't know labels!



PCA vs. Fisher Linear Discriminant



Principal Component Analysis

- higher variance
- bad for discriminability

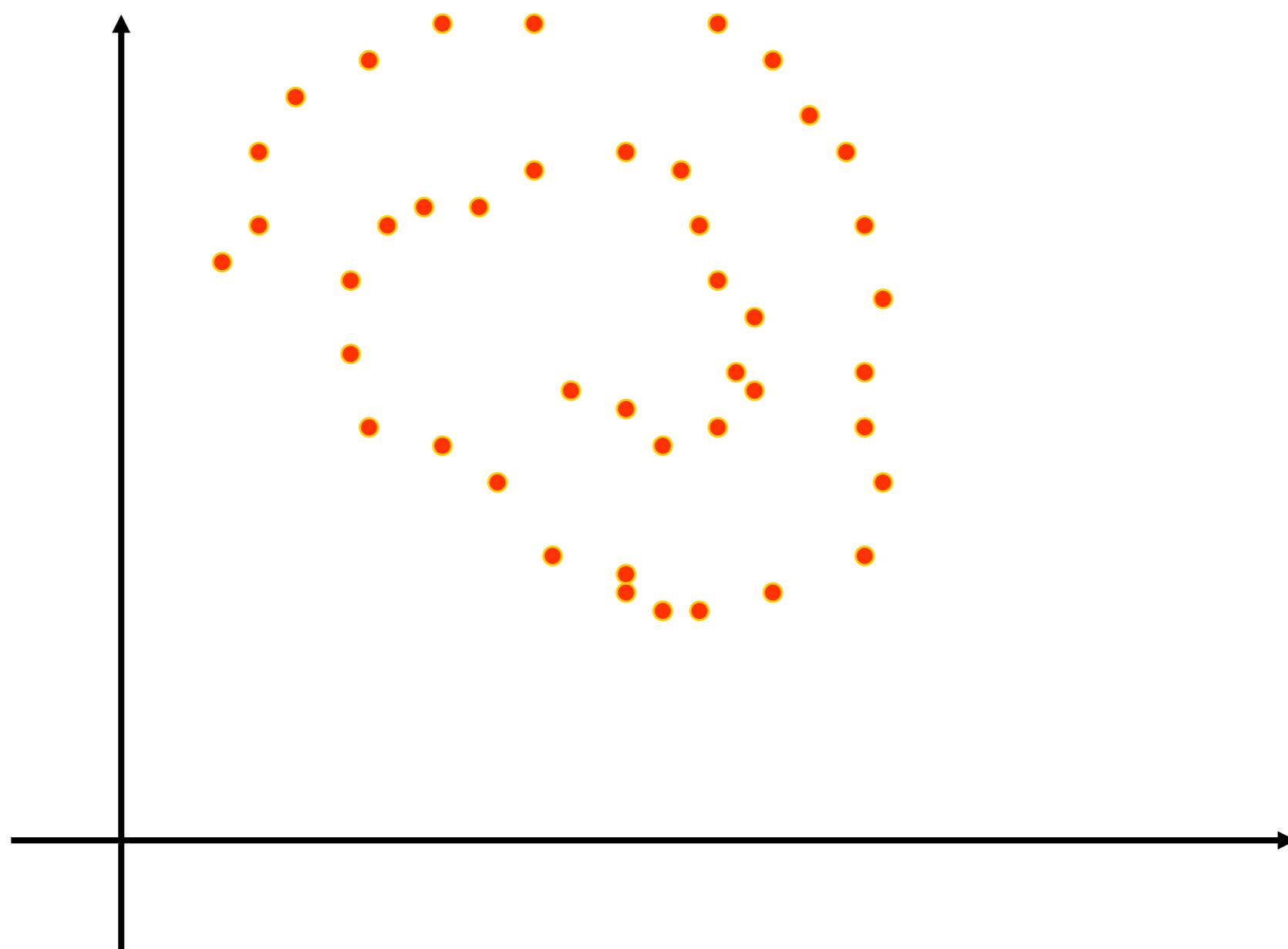
Fisher Linear Discriminant

- smaller variance
- good discriminability



Problematic Data Set for PCA

- PCA cannot capture NON-LINEAR structure!



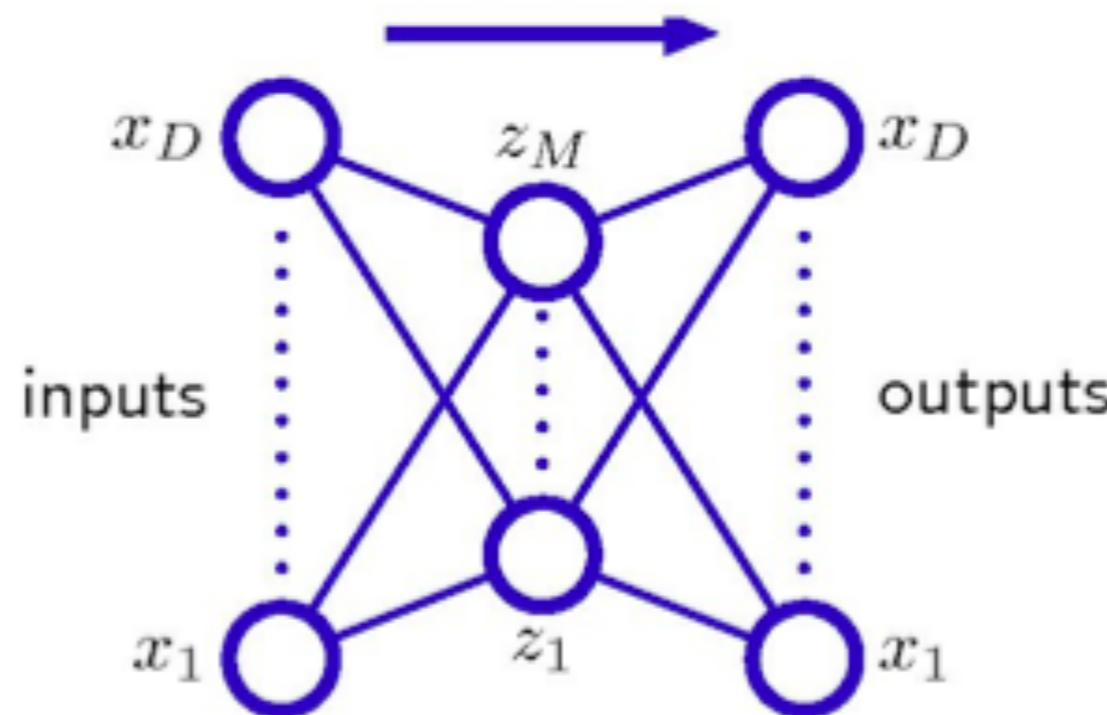
PCA Conclusions

- PCA
 - Finds orthonormal basis for data
 - Sorts dimensions in order of “importance”
 - Discard low significance dimensions
- Uses:
 - Get compact description
 - Ignore noise
 - Improve classification (hopefully)
- Not magic:
 - Doesn’t know class labels
 - Can only capture linear variations
- One of many tricks to reduce dimensionality!

Autoencoders

Relation to Neural Networks

- PCA is closely related to a particular form of neural network
- An **autoencoder** is a neural network whose outputs are its own inputs



- The goal is to minimize **reconstruction error**

Auto encoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

Auto encoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{W,V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}\|^2$$

Auto encoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{W,V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}\|^2$$

- If g and f are linear

$$\min_{W,V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - V W \mathbf{x}^{(n)}\|^2$$

Auto encoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{W,V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}\|^2$$

- If g and f are linear

$$\min_{W,V} \frac{1}{2N} \sum_{n=1}^N \|\mathbf{x}^{(n)} - V W \mathbf{x}^{(n)}\|^2$$

- In other words, the optimal solution is PCA

Auto encoders: Nonlinear PCA

- What if $g()$ is not linear?
- Then we are basically doing **nonlinear PCA**
- Some subtleties but in general this is an accurate description

Comparing Reconstructions



Real data

30-d deep autoencoder

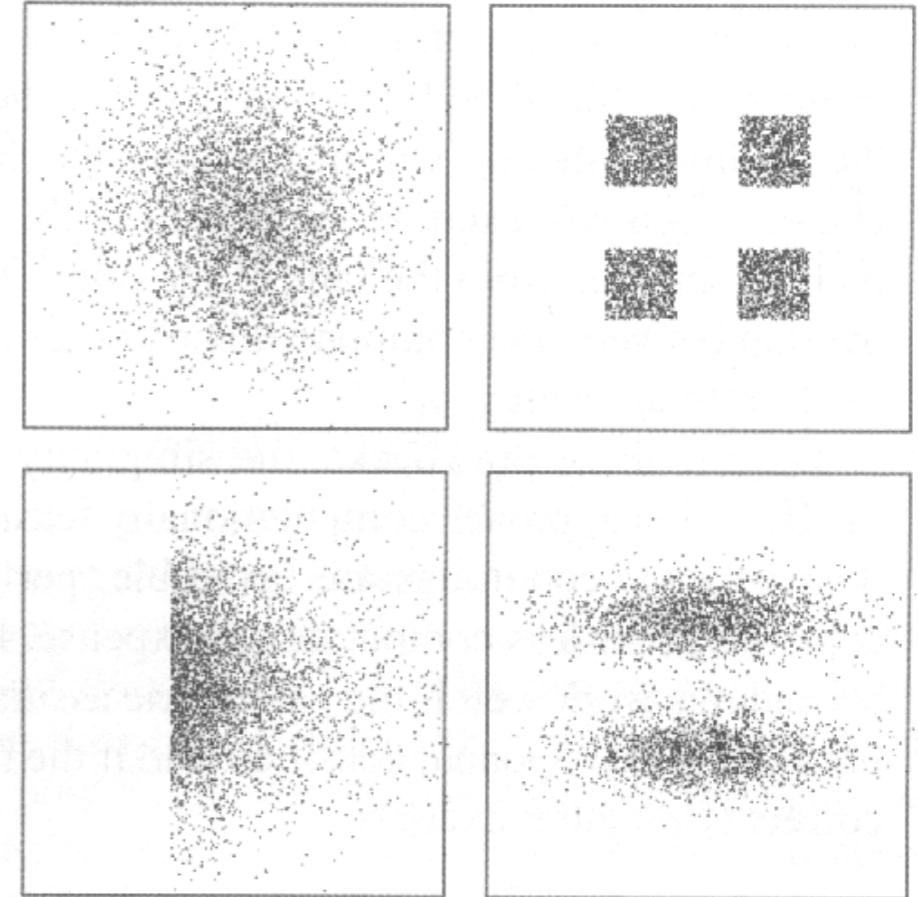
30-d logistic PCA

30-d PCA

Independent Component Analysis (ICA)

A Serious Limitation of PCA

- Recall that PCA looks at the covariance matrix only.
What if the data is not well described by the covariance matrix?



- The only distribution which is uniquely specified by its covariance (with the subtracted mean) is the Gaussian distribution. Distributions which deviate from the Gaussian are poorly described by their covariances.

Faithful vs Meaningful Representations

- Even with non-Gaussian data, variance maximization leads to the most faithful representation in a reconstruction error sense (recall that we trained our autoencoder network using a mean-square error in an input reconstruction layer).
- The mean-square error measure implicitly assumes Gaussianity, since it penalizes datapoints close to the mean less than those that are far away.
- But it does not in general lead to the most meaningful representation.
- We need to perform gradient descent in some function other than the reconstruction error.

A Criterion Stronger than Decorrelation

- The way to circumvent these problems is to look for components which are **statistically independent**, rather than just uncorrelated.
- For statistical independence, we require that

$$p(\xi_1, \xi_2, \dots, \xi_N) = \prod_{i=1}^N p(\xi_i)$$

- For uncorrelatedness, all we required was that

$$\langle \xi_i \xi_j \rangle - \langle \xi_i \rangle \langle \xi_j \rangle = 0, \quad i \neq j$$

- Independence is a stronger requirement; under independence,

$$\langle g_1(\xi_i) g_2(\xi_j) \rangle - \langle g_1(\xi_i) \rangle \langle g_2(\xi_j) \rangle = 0, \quad i \neq j$$

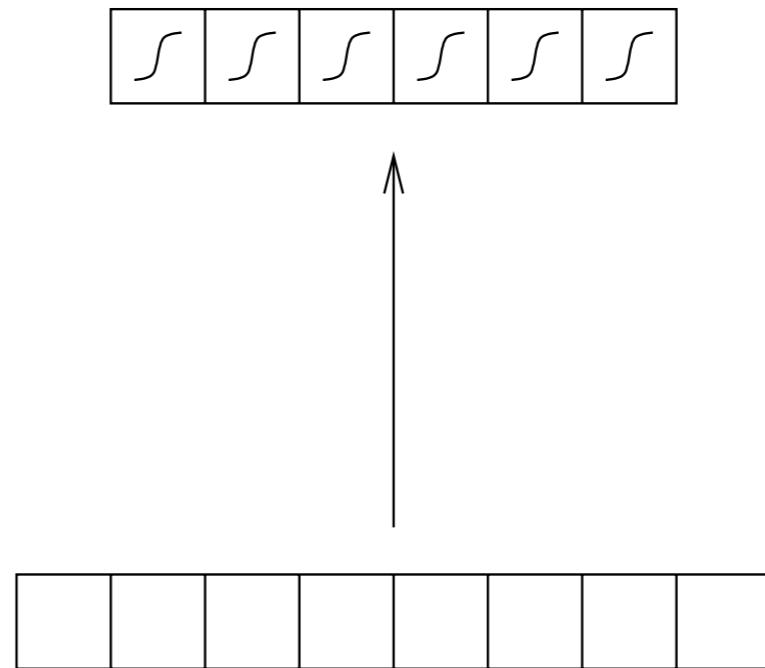
for any functions g_1 and g_2 .

Independent Component Analysis (ICA)

- Like PCA, except that we're looking for a transformation subject to the stronger requirement of independence, rather than uncorrelatedness.
- In general, no analytic solution (like eigenvalue decomposition for PCA) exists, so ICA is implemented using neural network models.
- To do this, we need an architecture and an objective function to descend/climb in.
- Leads to N independent (or as independent as possible) components in N -dimensional space; they need not be orthogonal.
- When are independent components identical to uncorrelated (principal) components? When the generative distribution is uniquely determined by its first and second moments. This is true of only the Gaussian distribution.

Neural Network for ICA

- Single layer network:

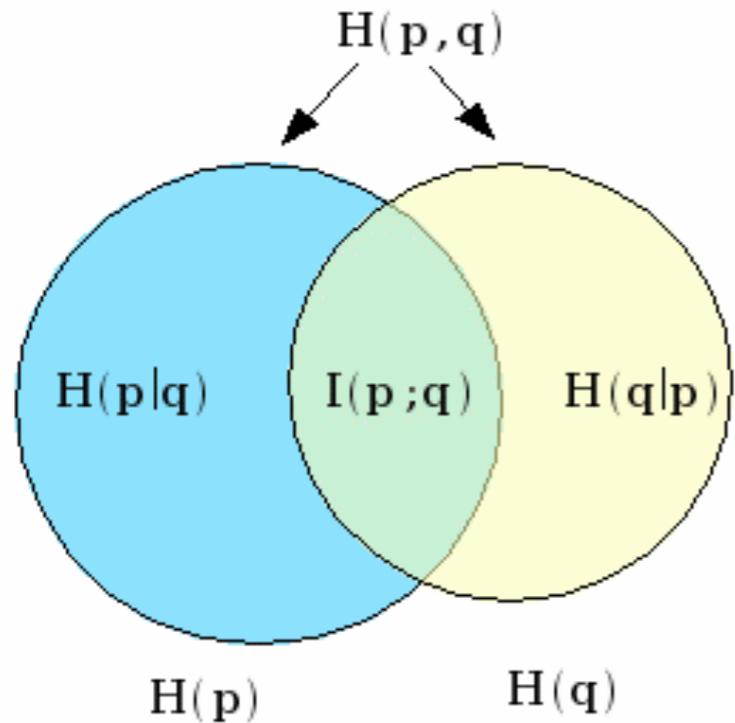


- Patterns $\{\xi\}$ are fed into the input layer.
- Inputs multiplied by weights in matrix \mathbf{W} .
- Output logistic (vector notation here):

$$\bar{y} = \frac{1}{1 + e^{\mathbf{W}^T \xi}}$$

Objective Function for ICA

- Want to ensure that the outputs y_i are maximally independent.
- This is identical to requiring that the mutual information be small. Or alternately that the joint entropy be large.



$H(p)$ = entropy of distribution p of first neuron's output

$H(p|q)$ = conditional entropy

$$\begin{aligned} I(p; q) &= H(p) - H(q|p) \\ &= H(q) - H(p|q) \\ &= \text{mutual information} \end{aligned}$$

- Gradient ascent in this objective function is called infomax (we're trying to maximize the enclosed area representing information quantities).

Blind Source Separation (BSS)

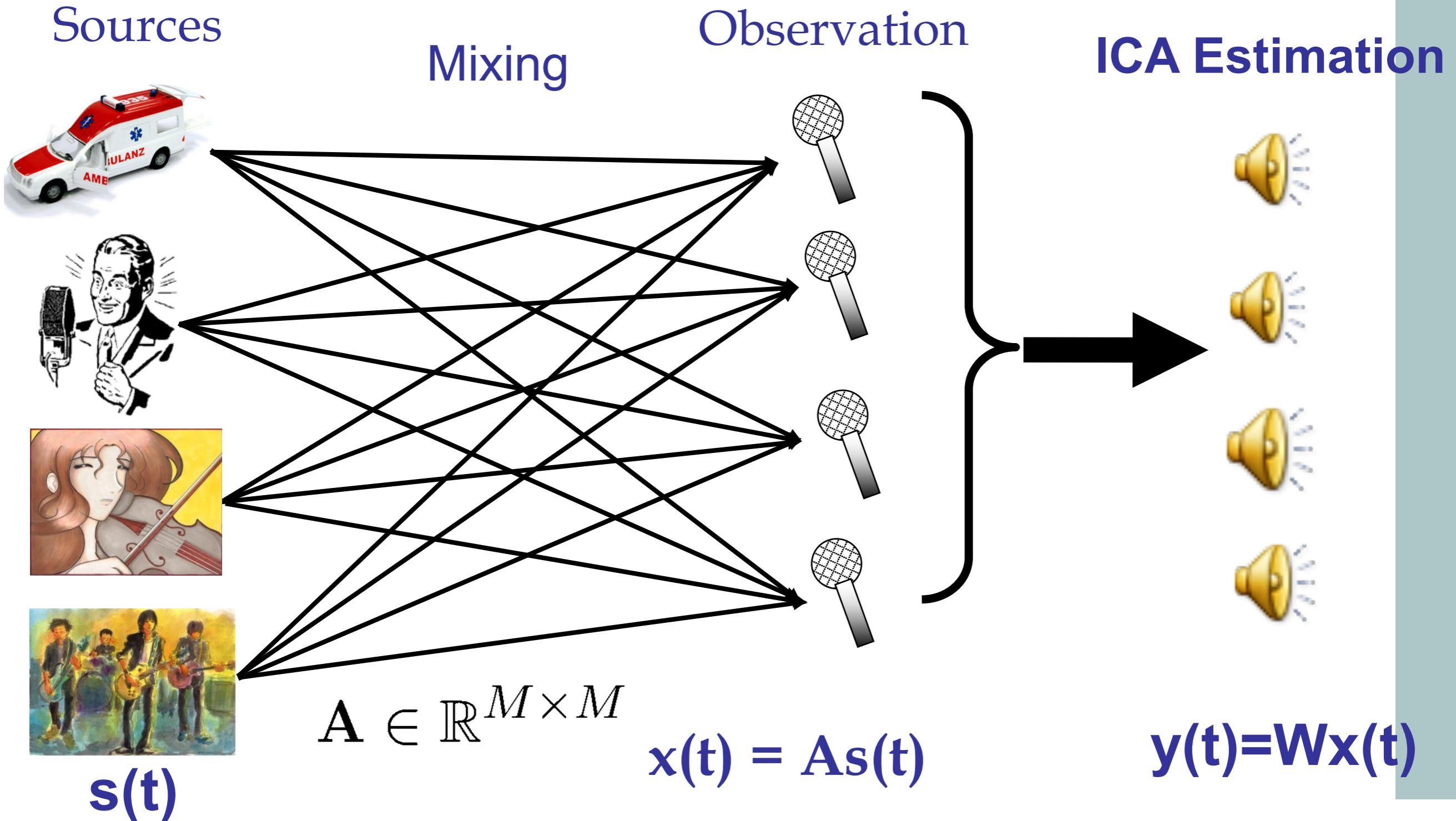
- The most famous application of ICA.
- Have K sources $\{s_k[t]\}$, and K signals $\{x_k[t]\}$. Both $\{s_k[t]\}$ and $\{x_k[t]\}$ are time series (t is a discrete time index).
- Each signal is a linear mixture of the sources

$$x_k[t] = \mathbf{A}s_k[t] + n_k[t]$$

where $n_k[t]$ is the noise contribution in the k th signal $x_k[t]$, and \mathbf{A} is a mixture matrix.

- The problem: given $x_k[n]$, determine \mathbf{A} and $s_k[n]$.

The Cocktail Party



Demo: The Cocktail Party

- Frequency domain ICA (1995)

Input mix:



Extracted speech:

