10-701: Introduction to Machine Learning Lecture 15: Dimensionality Reduction

Pradeep Ravikumar & Geoff Gordon

Spring 2026

Front Matter

- Announcements
  - TA project proposals have been turned in; all teams should schedule a time to have a 15-30 minute meeting with their TA mentor before Friday March 20!
  - HW4 has been released and is due on Tuesday, March 24
- Recommended Readings
  - Murphy, Chapters 12.2.1 - 12.2.3
  - Daumé III, Chapter 15: Unsupervised Learning

Unsupervised Learning

- Clustering: split an unlabeled data set into groups or partitions of “similar” data points
  - Use cases:
    - Organizing data
    - Discovering patterns or structure
    - Preprocessing for downstream tasks
- Dimensionality Reduction: given some unlabeled data set, learn a latent (typically lower-dimensional) representation
  - Use cases:
    - Decreasing computational costs
    - Improving generalization
    - Visualizing data

Data Visualization

Example:

Given 53 blood samples (features) from 65 people.

How can we visualize the measurements?

Data Visualization

Instances

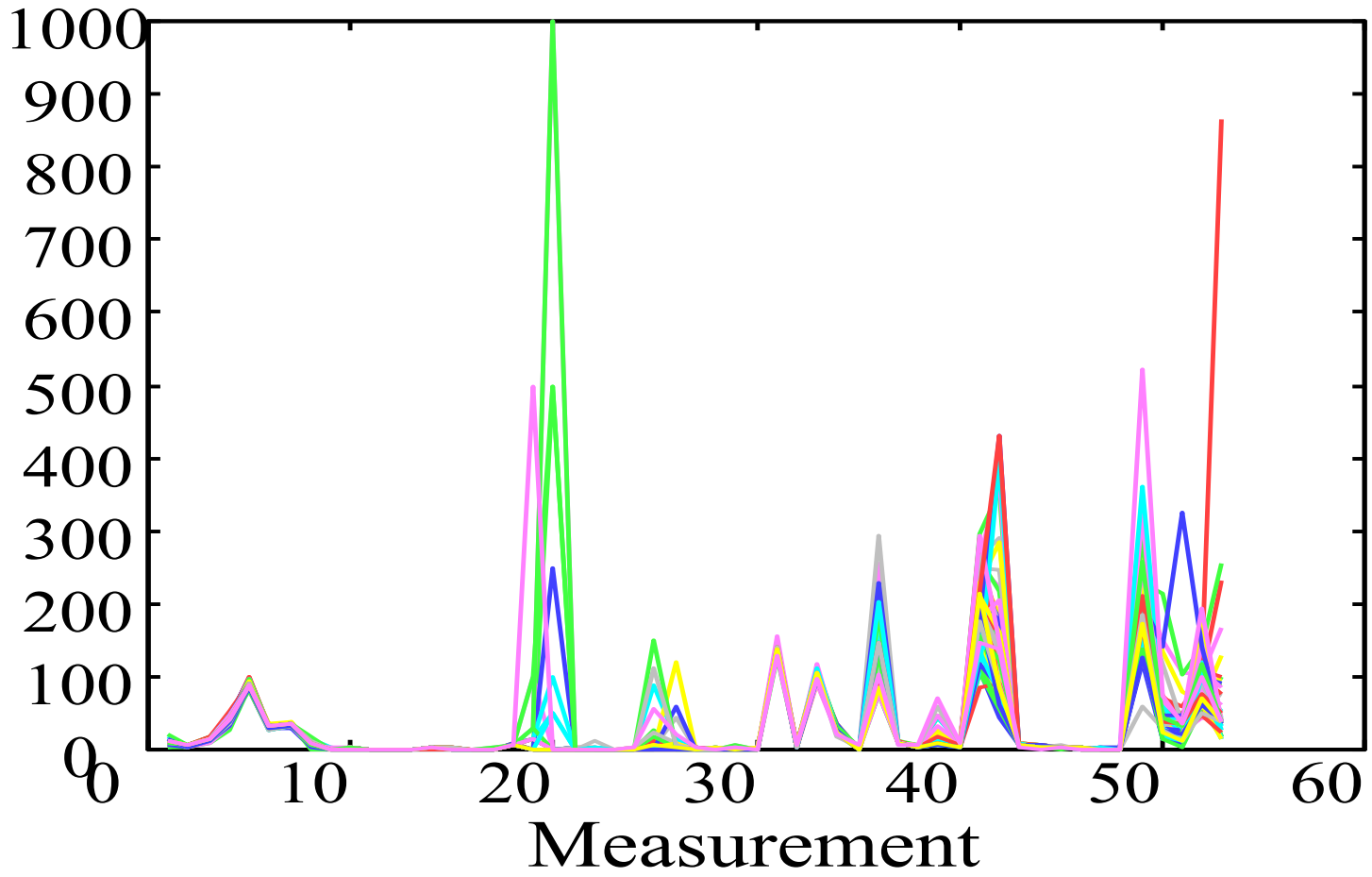
	H-WBC	H-RBC	H-Hgb	H-Hct	H-MCV	H-MCH	H-MCHC
A1	8.0000	4.8200	14.1000	41.0000	85.0000	29.0000	34.0000
A2	7.3000	5.0200	14.7000	43.0000	86.0000	29.0000	34.0000
A3	4.3000	4.4800	14.1000	41.0000	91.0000	32.0000	35.0000
A4	7.5000	4.4700	14.9000	45.0000	101.0000	33.0000	33.0000
A5	7.3000	5.5200	15.4000	46.0000	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

Difficult to see the correlations between the features...

Data Visualization

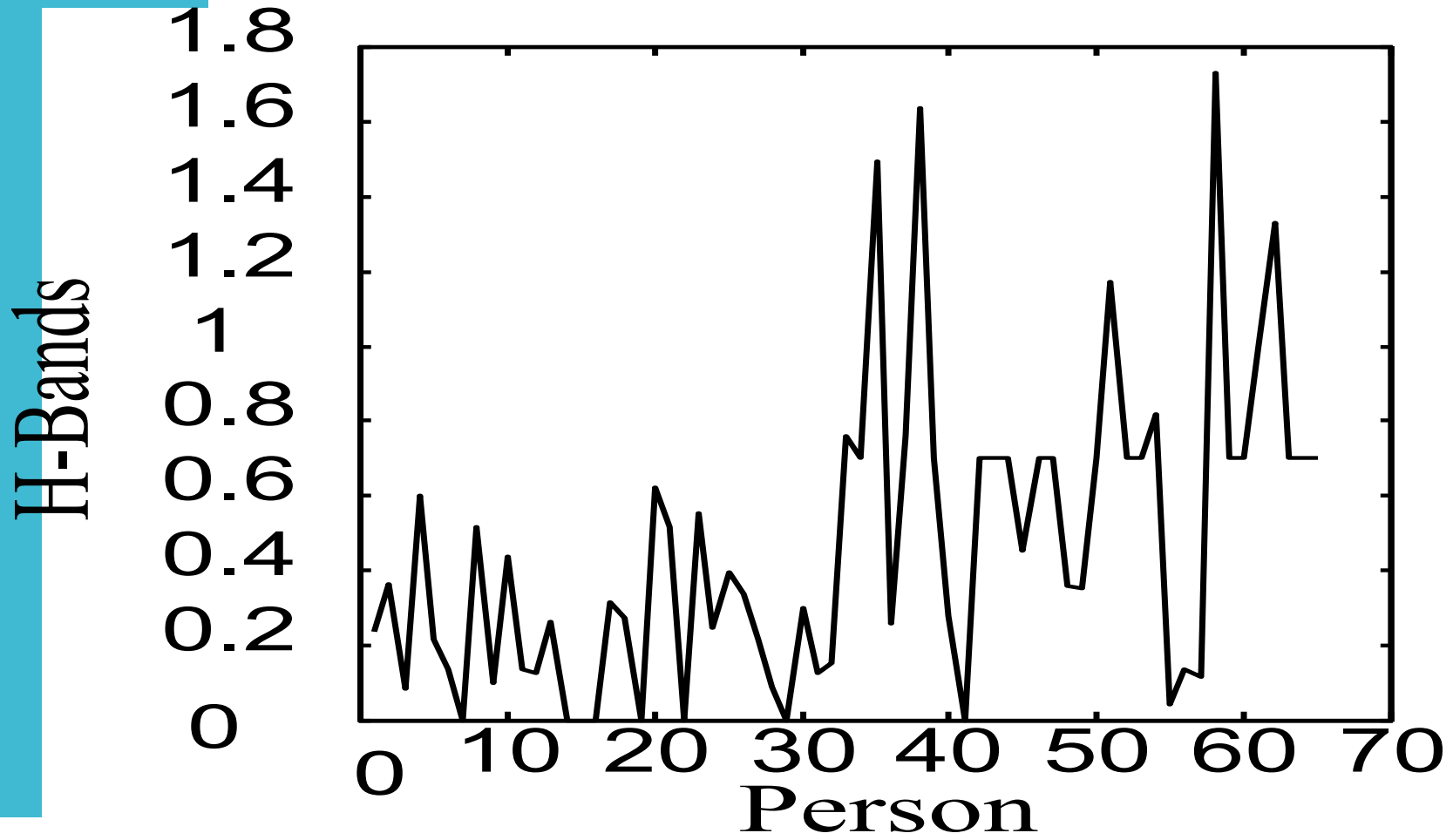
Curves (65 curves, one for each person)



Difficult to compare the different patients...

Data Visualization

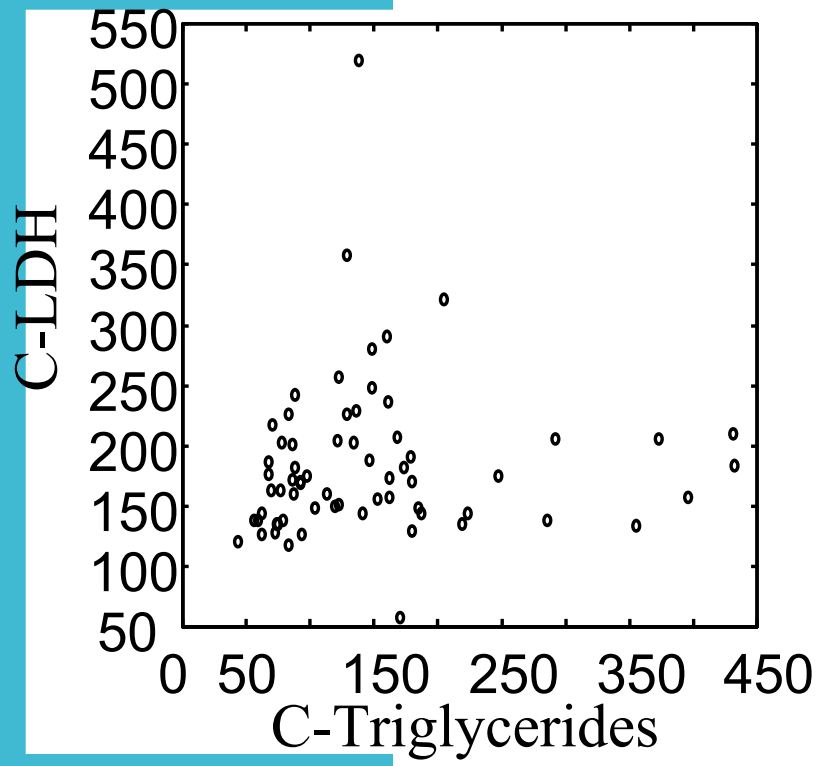
- Curves (53 pictures, one for each feature)



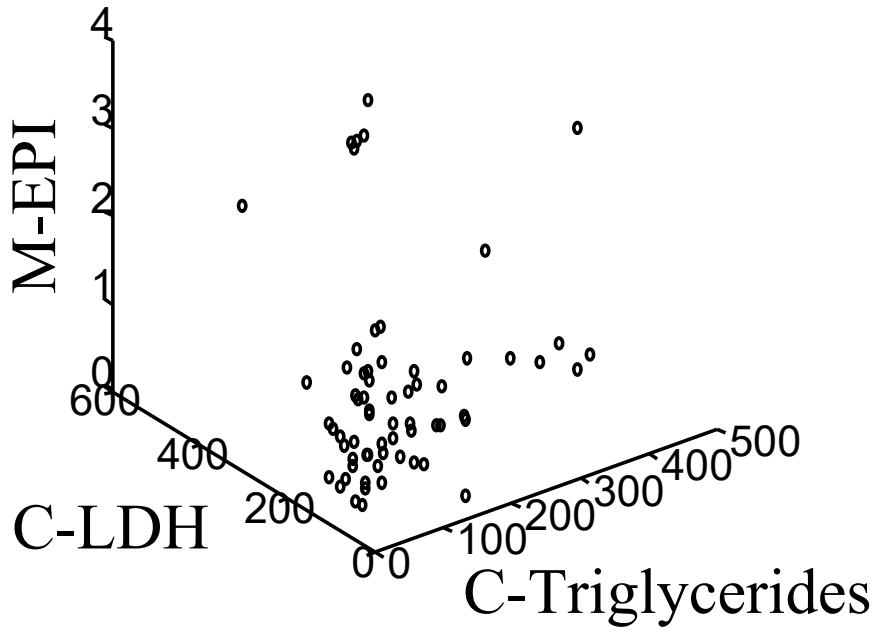
Difficult to see the correlations between the features...

Data Visualization

Bi-variate



Tri-variate

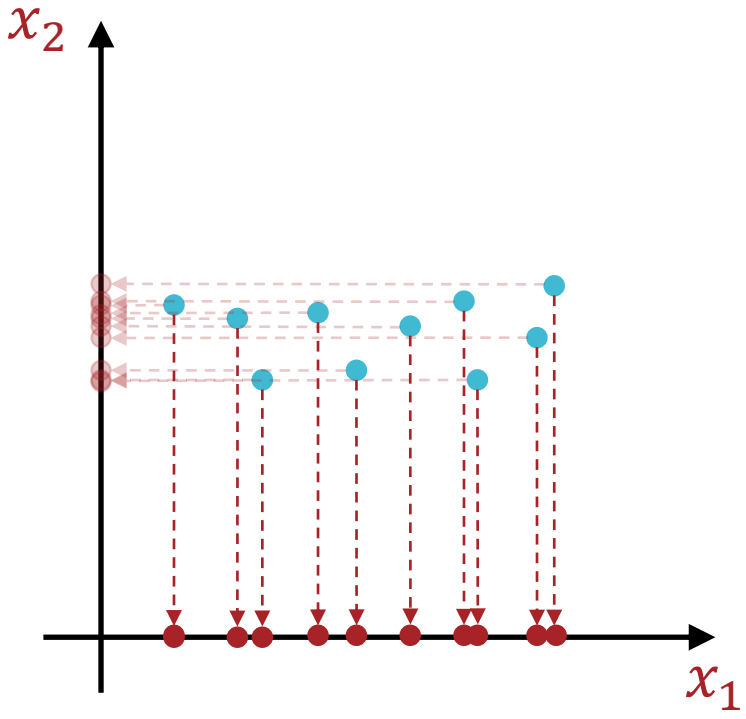


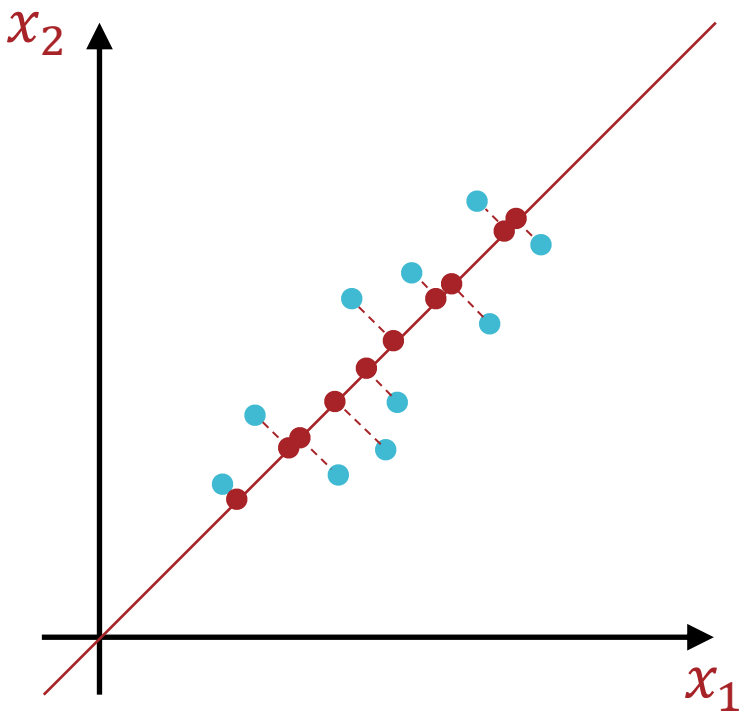
How can we visualize the other variables???

… difficult to see in 4 or higher dimensional spaces…

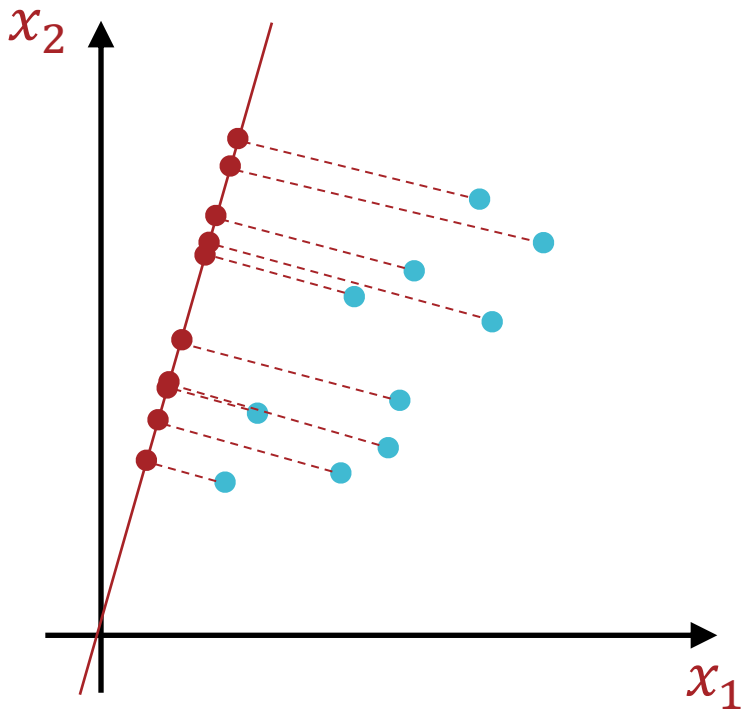
Data Visualization

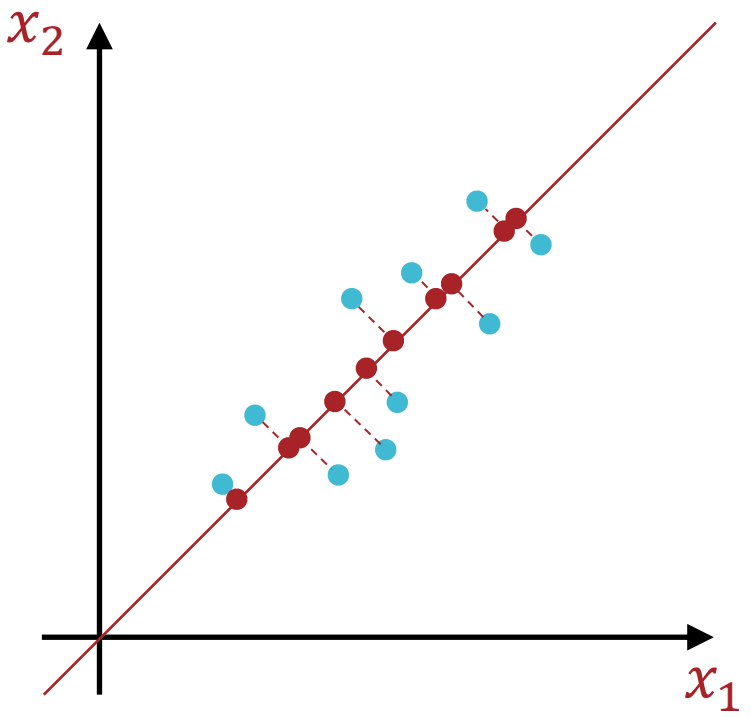
- Is there a representation better than the coordinate axes?
- Is it really necessary to show all the 53 dimensions?
  - … what if there are strong correlations between the features?
- How could we find
  the *smallest* subspace of the 53-D space that
  keeps the *most information* about the original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



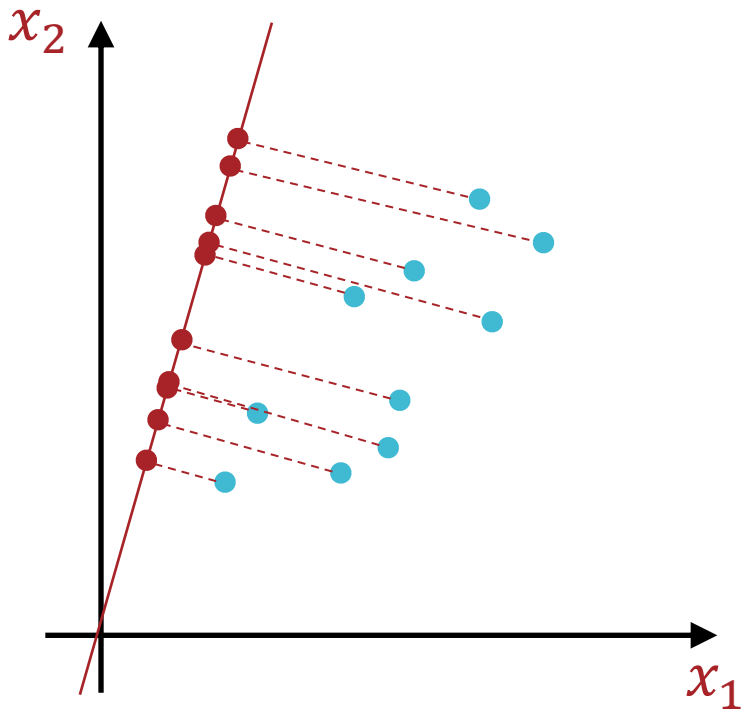


Feature Elimination

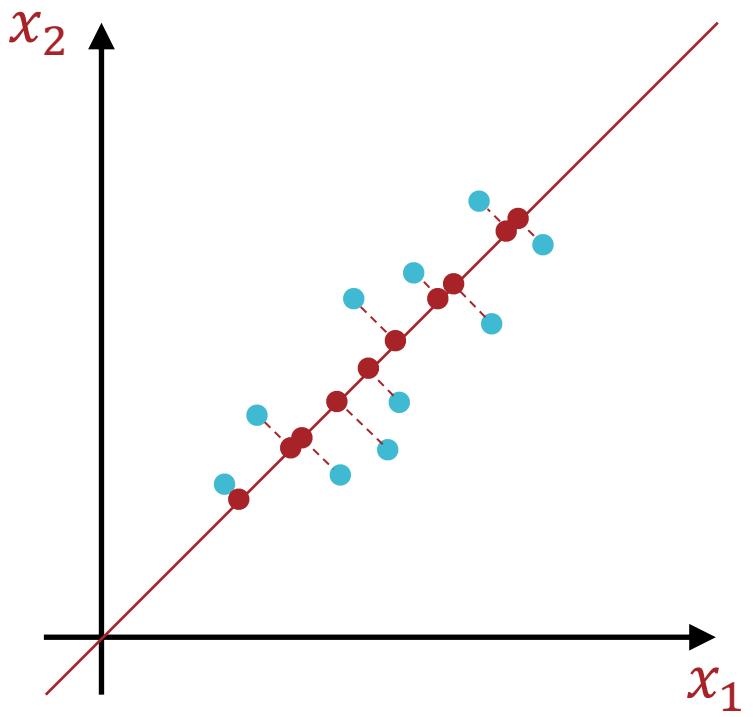




Feature Reduction



Option A



Option B

Which projection do you prefer?

Centering the Data

- To be consistent, we will constrain principal components to be *orthogonal unit vectors* that begin at the origin
- Preprocess data to be centered around the origin:

1. $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$
2. $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{\mu}\ \forall\ n$
3. $X = \begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)^T} \\ \tilde{\boldsymbol{x}}^{(2)^T} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(N)^T} \end{bmatrix}$

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(n)} = \left(\frac{\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}}{\|\boldsymbol{v}\|_2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Length of projection

Direction of projection

Reconstruction Error

- The projection of $\tilde{\boldsymbol{x}}^{(n)}$ onto a unit vector $\boldsymbol{v}$ is

$$\boldsymbol{z}^{(n)} = \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}$$

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$\left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$= \tilde{\boldsymbol{x}}^{(n)^T} \tilde{\boldsymbol{x}}^{(n)} - 2\left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)} + \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \boldsymbol{v}$$

$$= \tilde{\boldsymbol{x}}^{(n)^T} \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}$$

$$= \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

Minimizing the Reconstruction Error

⇕

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v} \right\|_2^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \sum_{n=1}^{N} \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

← Variance of projections ($\tilde{\boldsymbol{x}}^{(n)}$ are centered)

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T \left(\sum_{n=1}^{N} \tilde{\boldsymbol{x}}^{(n)} \tilde{\boldsymbol{x}}^{(n)^T} \right) \boldsymbol{v}$$

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X) \boldsymbol{v}$$

Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$\begin{aligned}\mathcal{L}(\boldsymbol{v}, \lambda) &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\|\boldsymbol{v}\|_2^2 - 1) \\ &= \boldsymbol{v}^T (X^T X)\boldsymbol{v} - \lambda(\boldsymbol{v}^T \boldsymbol{v} - 1)\end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = (X^T X)\boldsymbol{v} - \lambda \boldsymbol{v}$$

$$\rightarrow (X^T X)\hat{\boldsymbol{v}} - \lambda \hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda \hat{\boldsymbol{v}}$$

- $\hat{\boldsymbol{v}}$ is an eigenvector of $X^T X$ and λ is the corresponding eigenvalue! But which one?

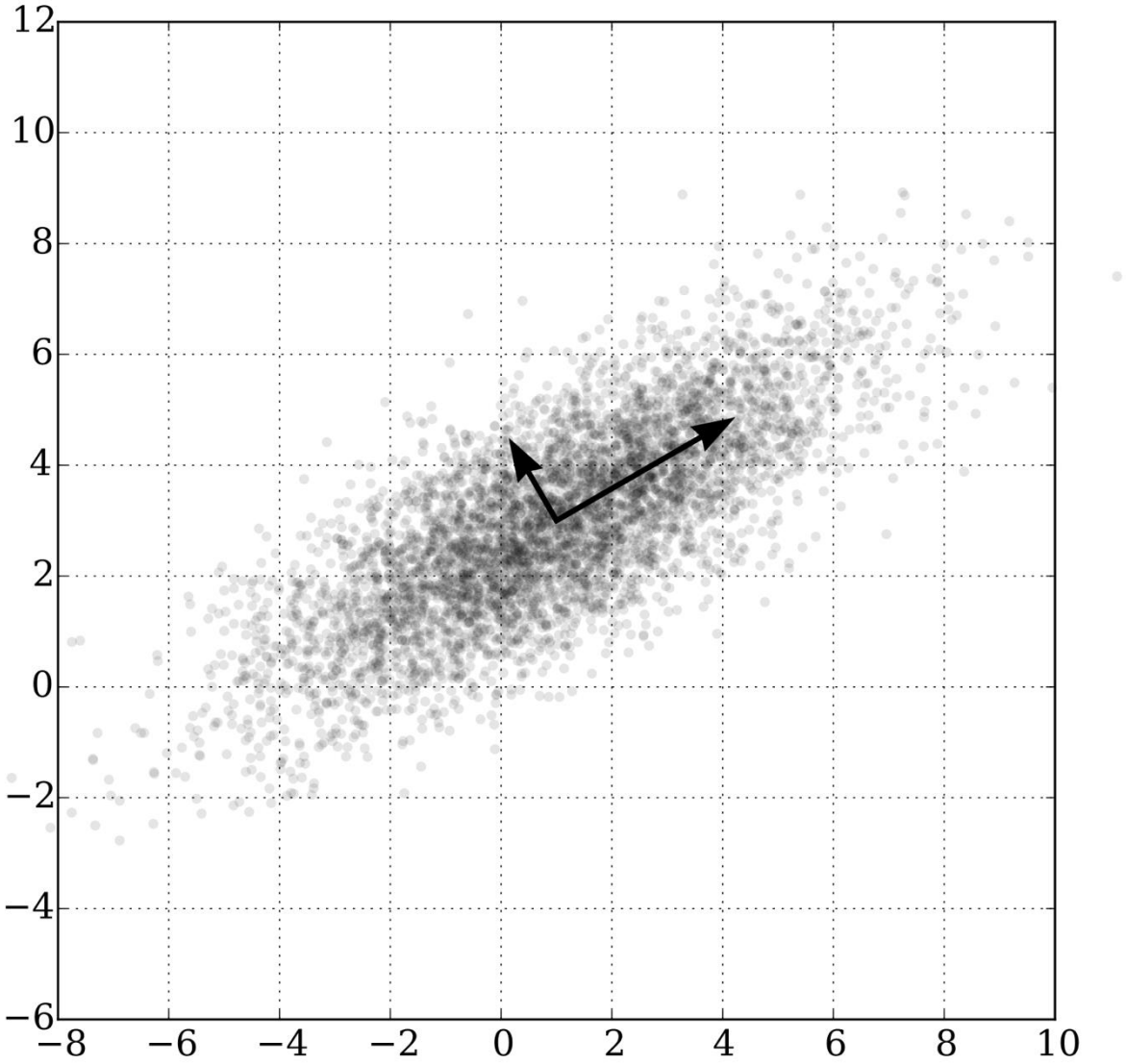
Maximizing the Variance

$$\hat{\boldsymbol{v}} = \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmax}} \boldsymbol{v}^T (X^T X)\boldsymbol{v}$$

$$(X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}} \rightarrow \hat{\boldsymbol{v}}^T (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}} = \lambda$$

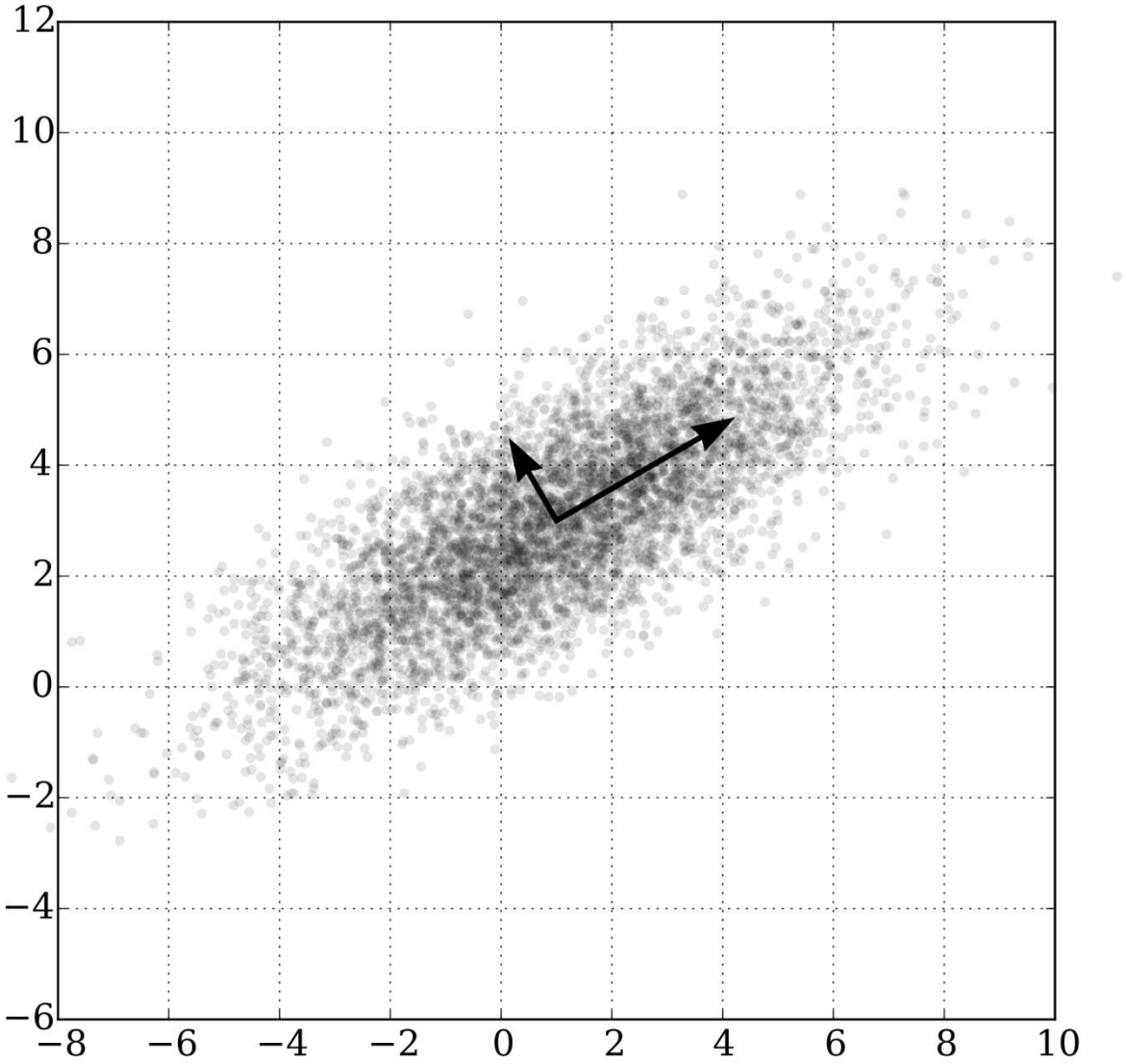
- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_1
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthogonal
- Etc …
- λ_i is a measure of how much variance falls along $\hat{\boldsymbol{v}}_i$

Principal Components: Example



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

How can we efficiently find principal components (eigenvectors)?



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

PCA algorithm II (sample covariance matrix)

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

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

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

PCA basis vectors = the eigenvectors of Σ

Larger eigenvalue $\Rightarrow$ more important eigenvectors

PCA algorithm II (sample covariance matrix)

PCA algorithm($\mathbf{X}$, k): top k eigenvalues/eigenvectors

% $\underline{\mathbf{X}}$ = N × m data matrix,
% … each data point $\mathbf{x}_i$ = column vector, i=1..m

- $\underline{\mathbf{x}} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}_i$
- $\mathbf{X}$ ← subtract mean $\underline{\mathbf{x}}$ from each column vector $\mathbf{x}_i$ in $\underline{\mathbf{X}}$
- $\Sigma \leftarrow \mathbf{X}\mathbf{X}^T$ … covariance matrix of $\mathbf{X}$
- $\{ \lambda_i, \mathbf{u}_i \}_{i=1..N}$ = eigenvectors/eigenvalues of Σ
  … $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$
- Return $\{ \lambda_i, \mathbf{u}_i \}_{i=1..k}$
  % top k PCA components

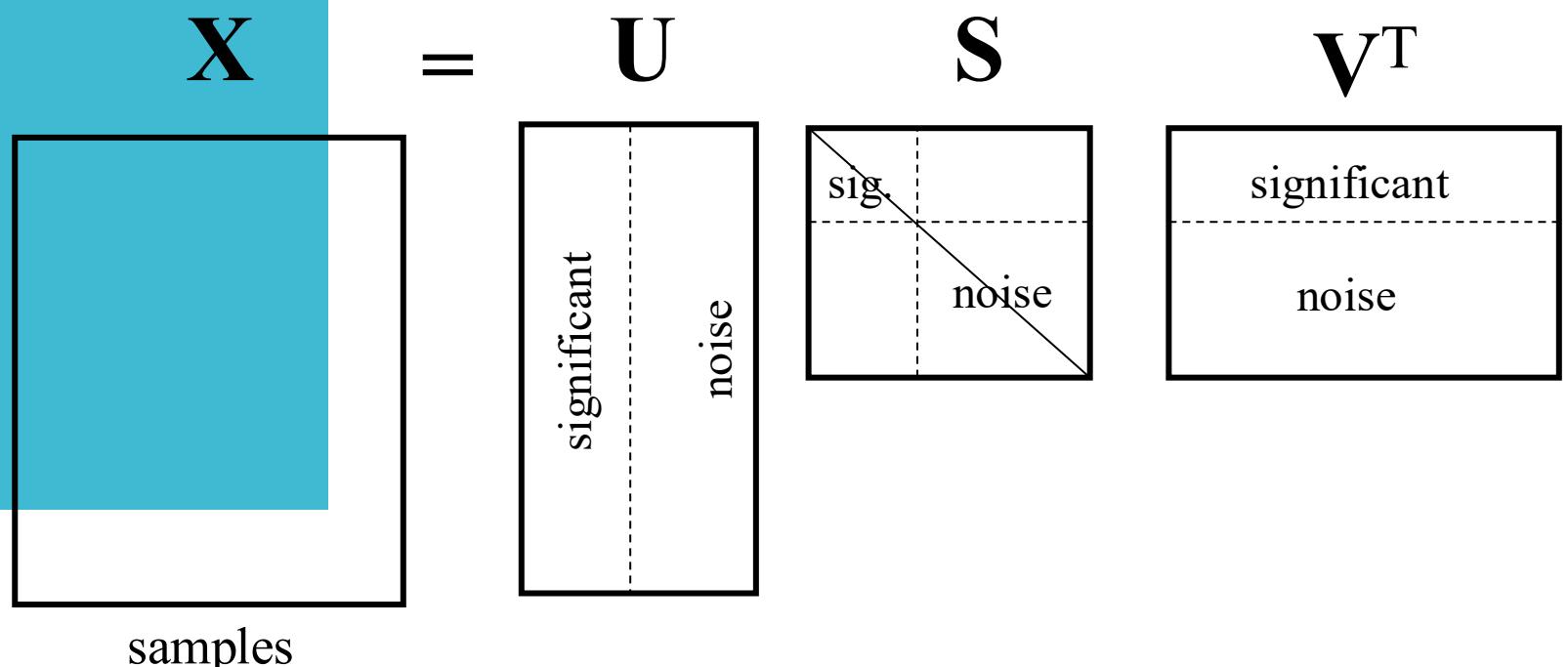
PCA algorithm III (SVD of the data matrix)

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

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



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

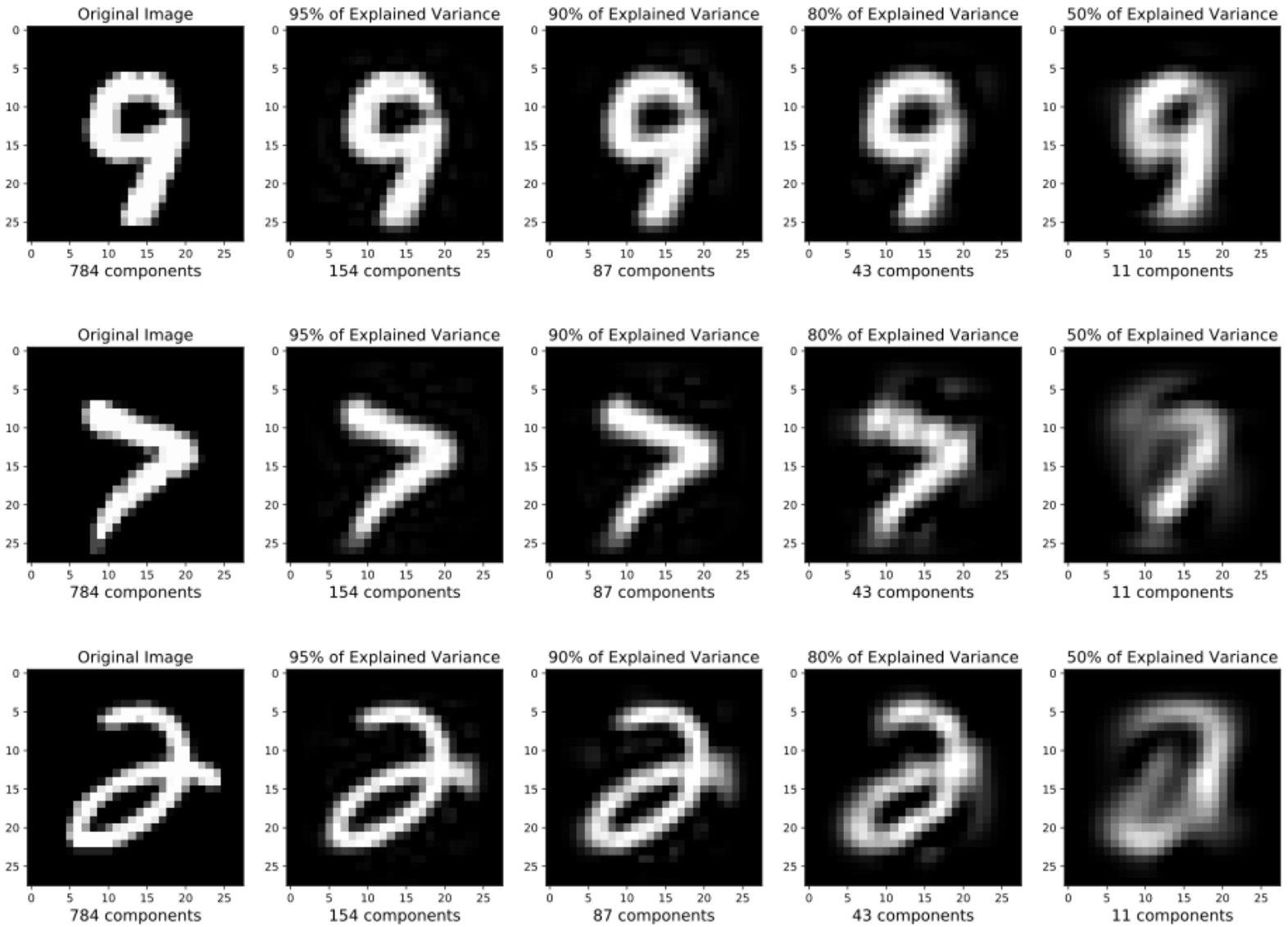
Choosing the number of PCs

- Define a percentage of explained variance for the i^{th} PC:

$$\lambda_i \Big/ \sum \lambda_j$$

- Select all PCs above some threshold of explained variance, e.g., 5%
- Keep selecting PCs until the total explained variance exceeds some threshold, e.g., 90%
- Evaluate on some downstream metric

PCA Example: MNIST Digits



Figures courtesy of Matt Gormley

PCA Example: MNIST Digits

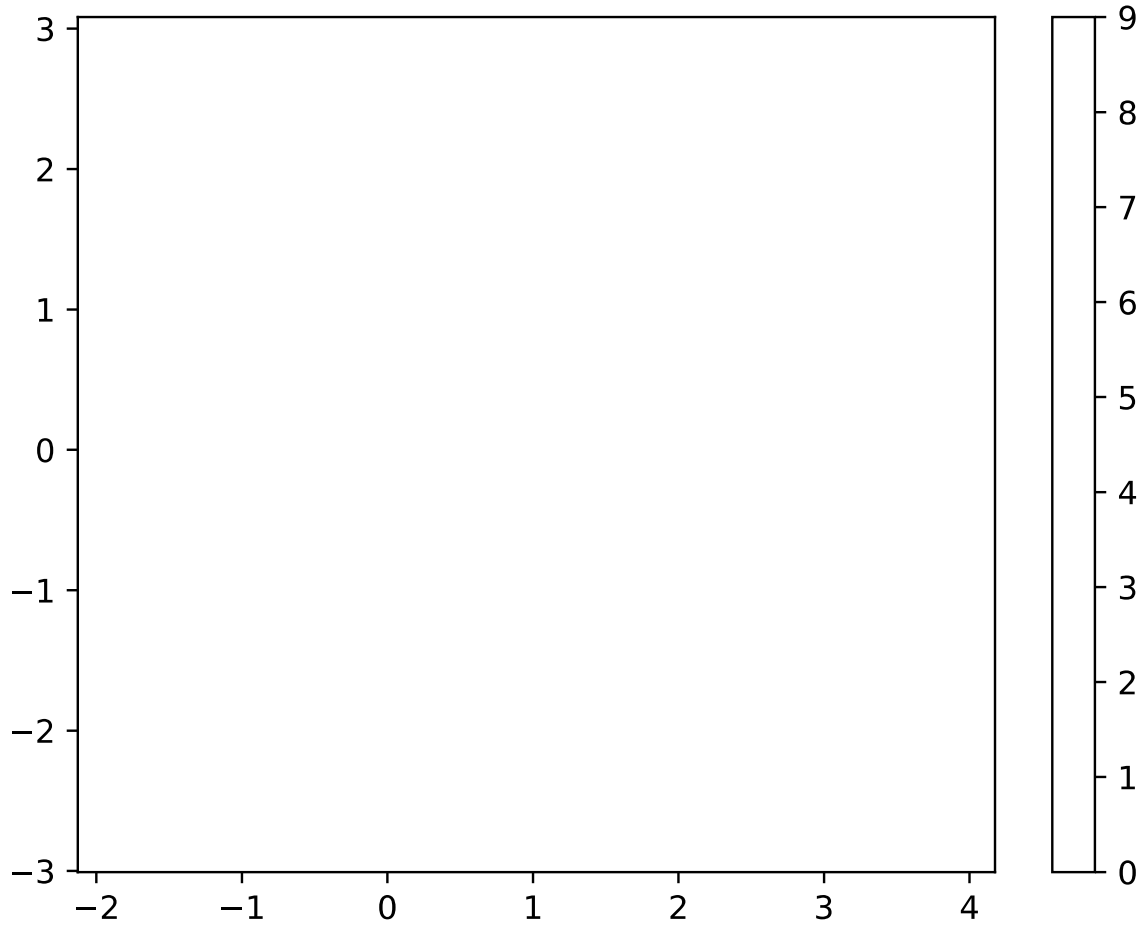


Figure courtesy of Matt Gormley

PCA Example: MNIST Digits

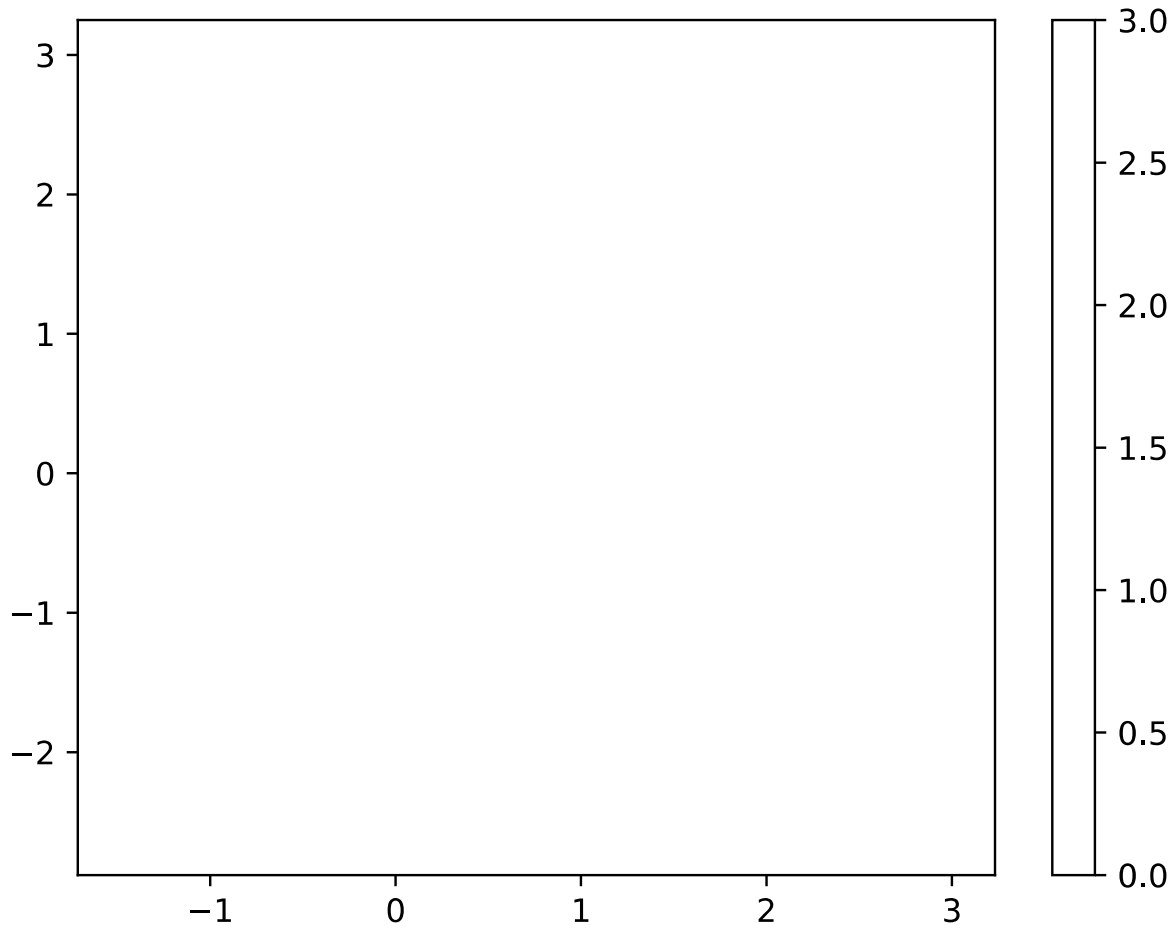


Figure courtesy of Matt Gormley

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

Applying PCA: Eigenfaces

Method: Use PCA on the *whole dataset* to get "principal component" images ("eigenfaces"), and then classify based on projection weights onto these principal component images

Applying PCA: Eigenfaces

Example data set: Images of faces

Eigenface approach
[Turk & Pentland], [Sirovich & Kirby]

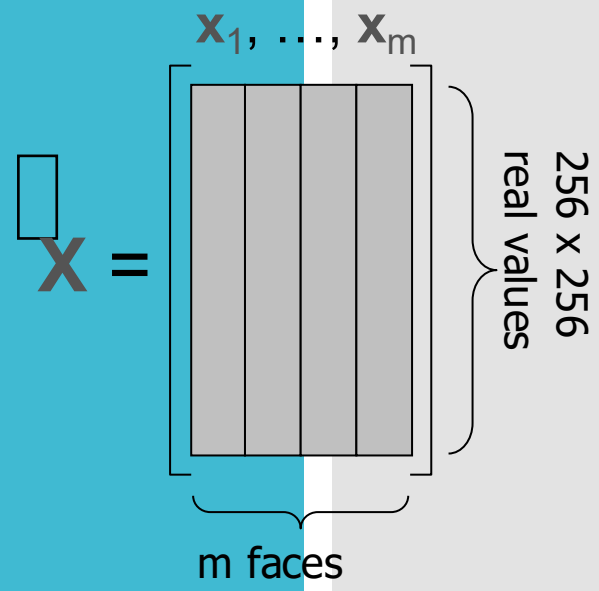
Each face **x** is …

256×256 values (luminance at location)

x in $\Re^{256\times256}$ (view as 64K dim vector)

Form $\mathbf{X} = [\, \mathbf{x}_1, \ldots, \mathbf{x}_m \,]$ **centered** data matrix

Compute $\Sigma = \mathbf{X}\mathbf{X}^T$



Principal Components

Reconstructing…

… faster if trained with…

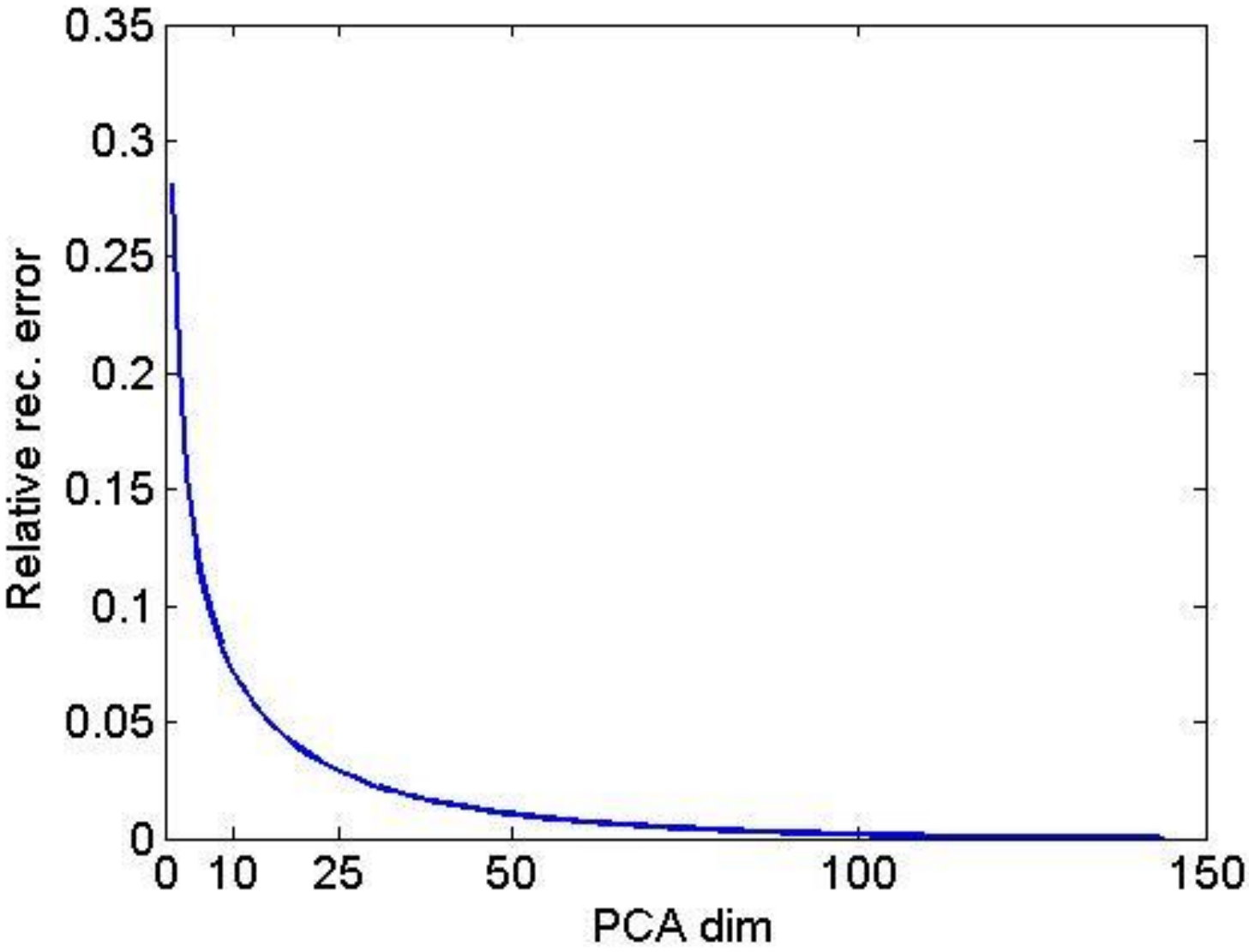
only people w/out glasses

- same lighting conditions

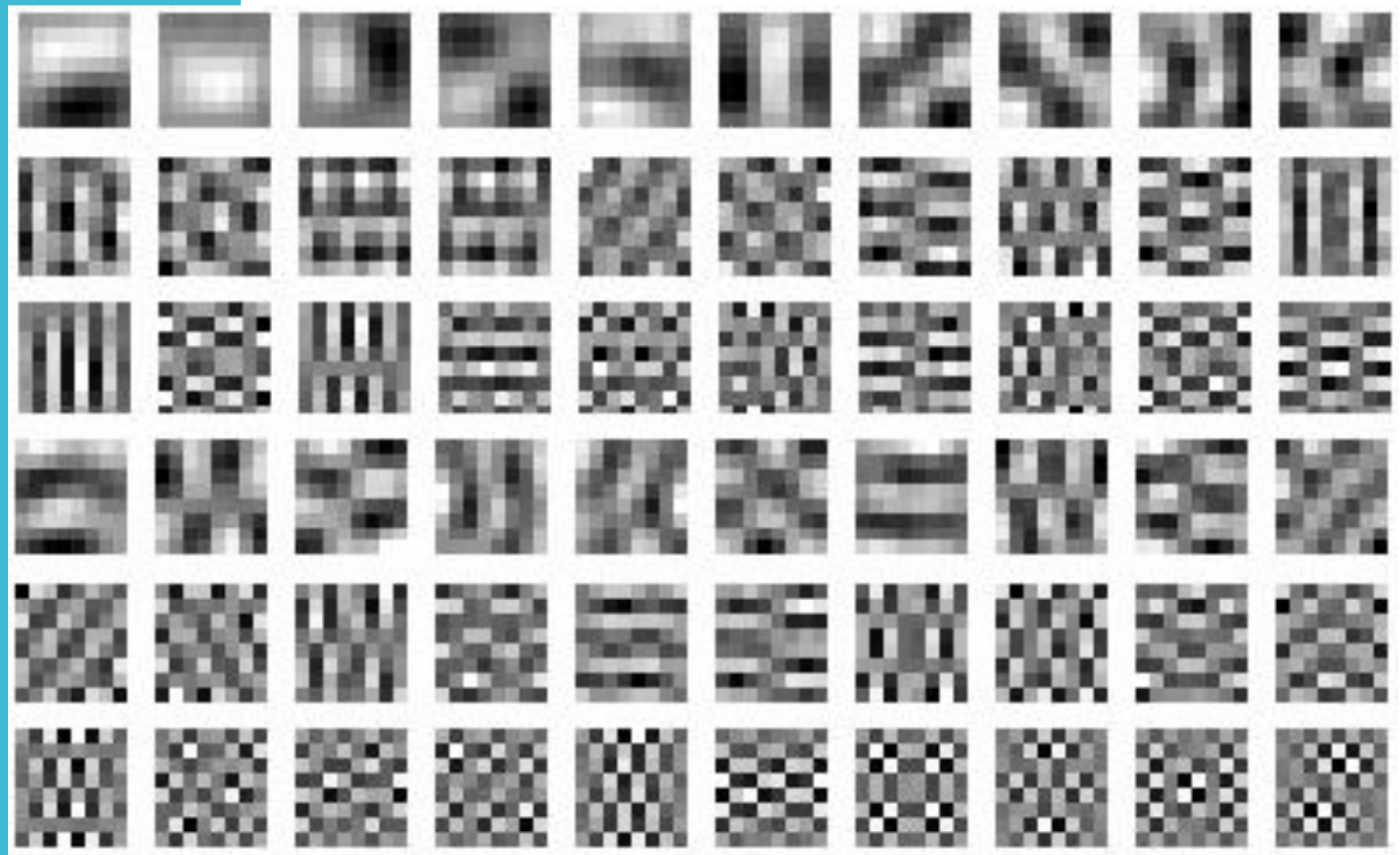
Original Image

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

L_2 error and PCA dim

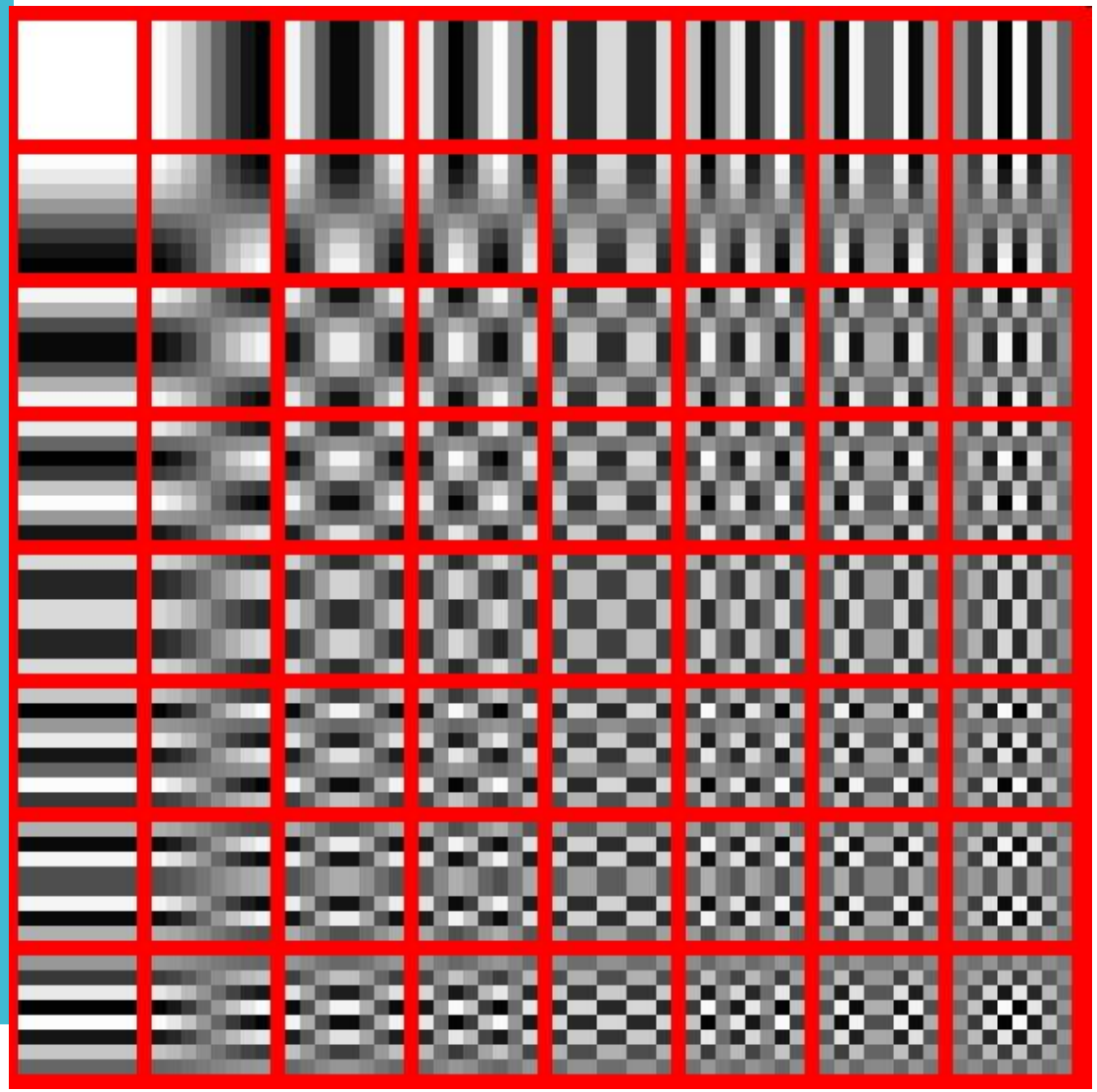


60 most important eigenvectors



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

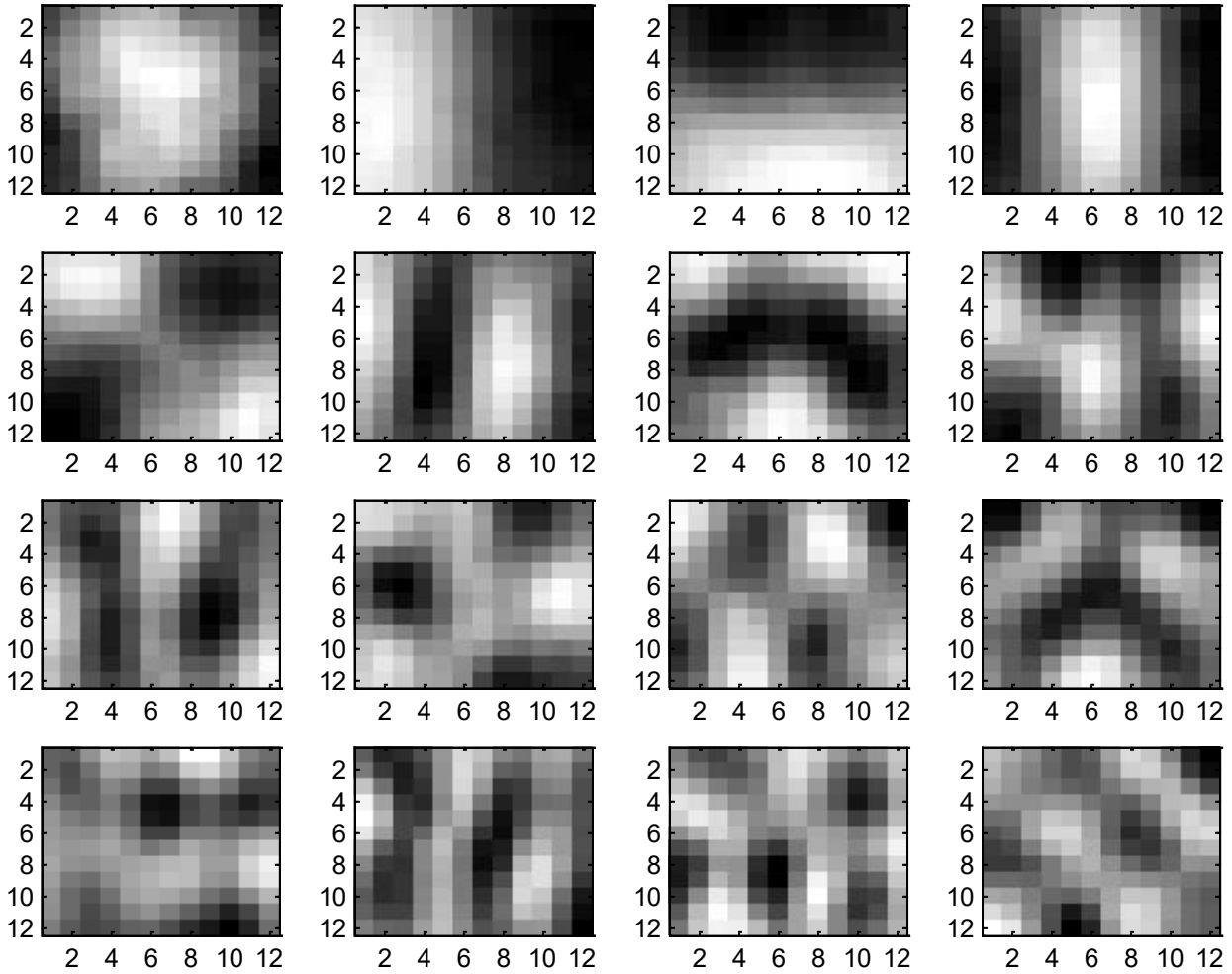
2D Discrete Cosine Basis



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

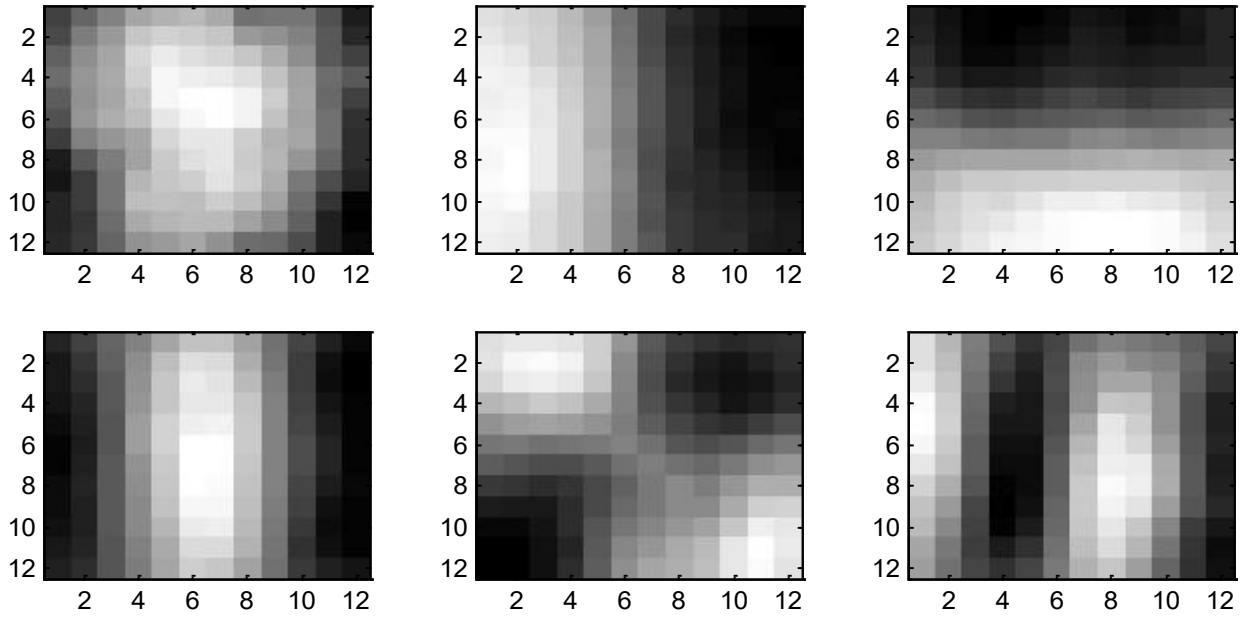
PCA compression: 144D → 60D

16 most important eigenvectors



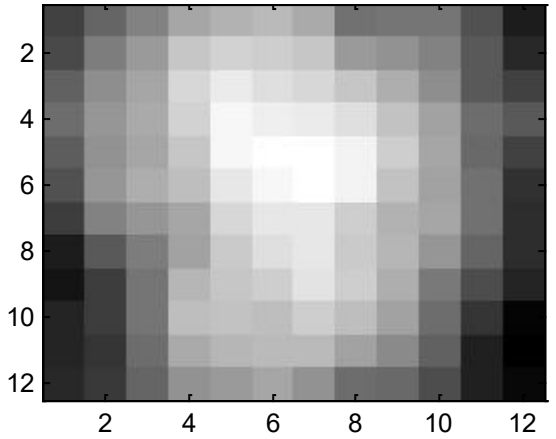
PCA compression: 144D → 16D

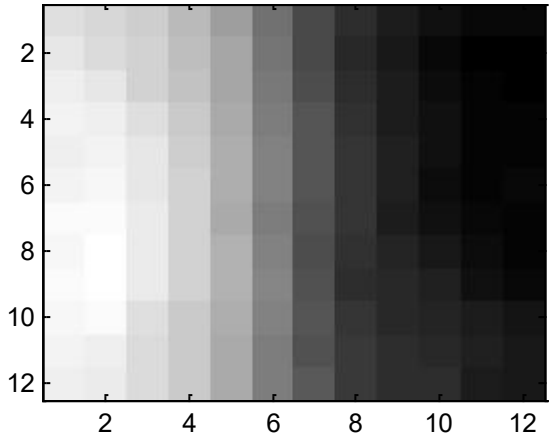
6 most important eigenvectors

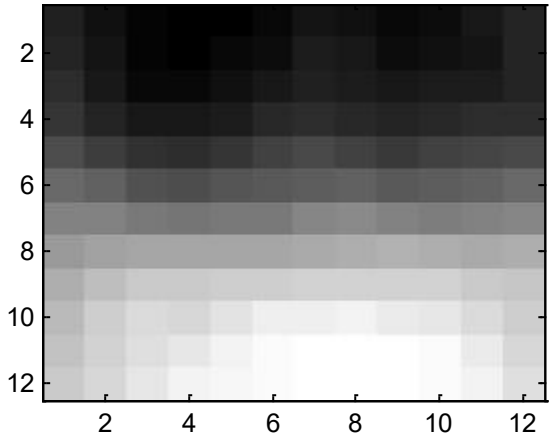


PCA compression: 144D → 6D

3 most important eigenvectors

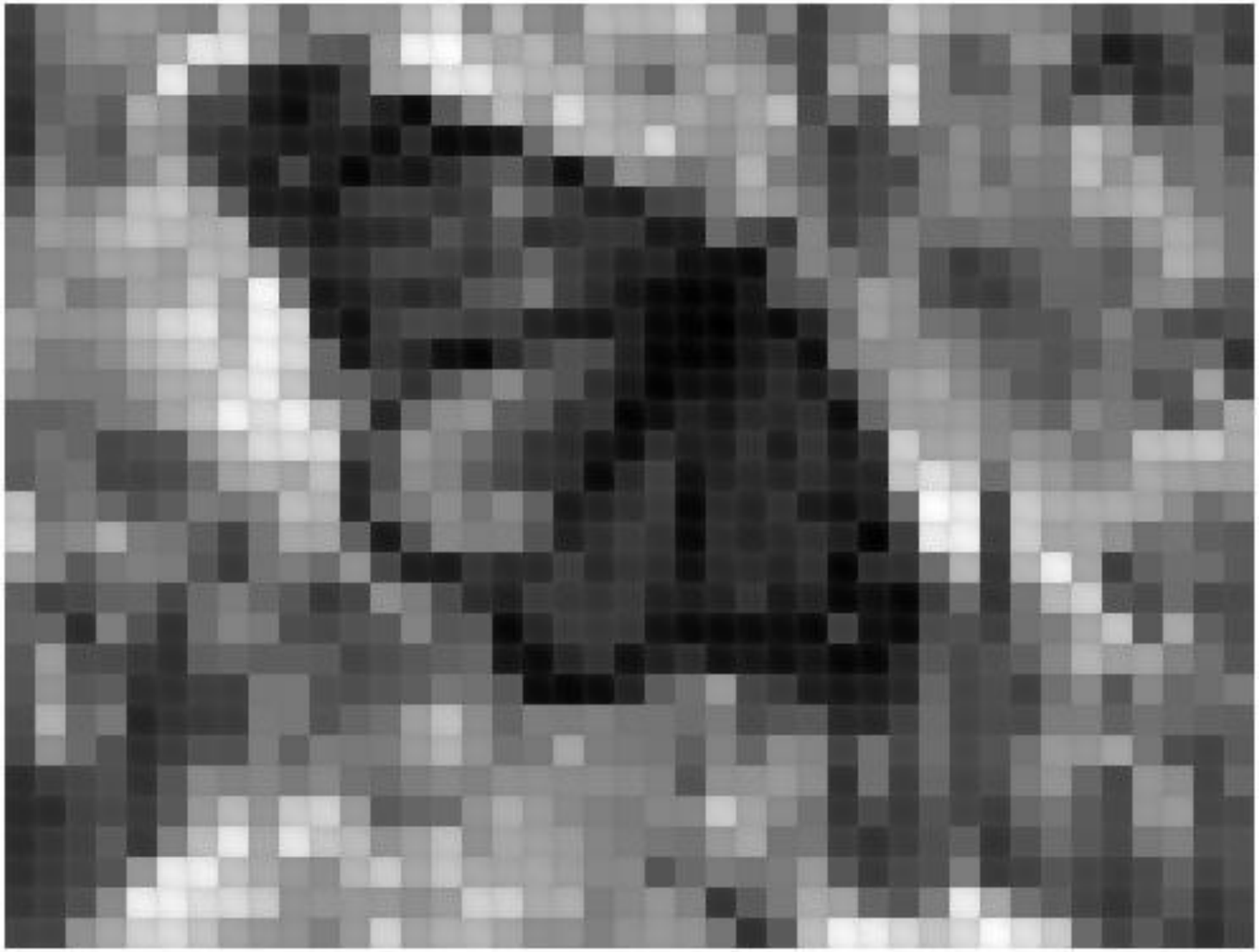




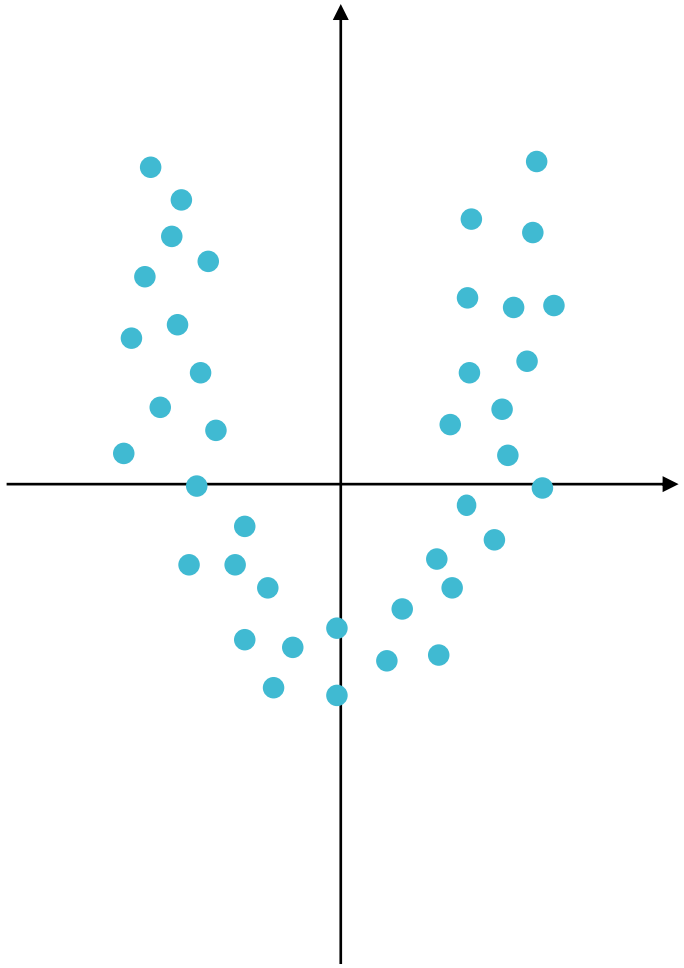


PCA compression: 144D → 3D

PCA compression: 144D → 1D

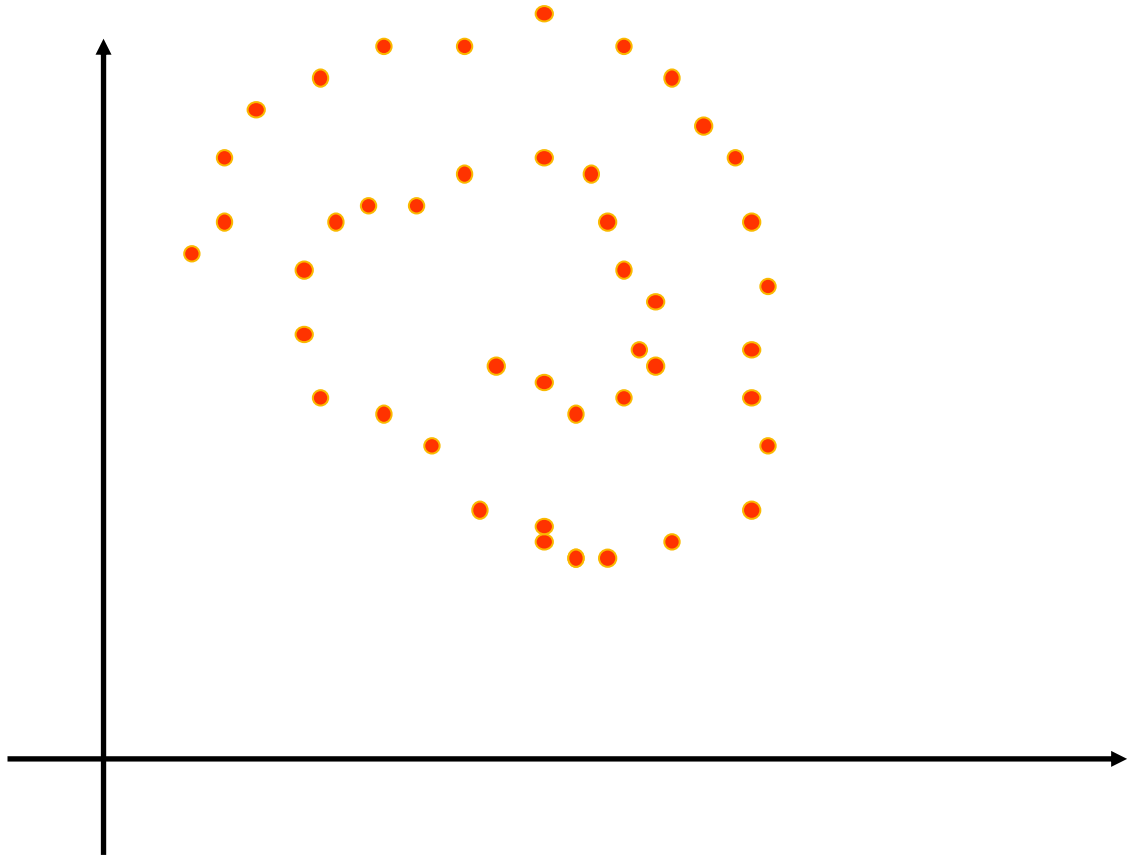


Shortcomings of PCA



- Principal components are orthogonal (unit) vectors
- Principal components can be expressed as linear combinations of the data

Problematic Data Set for PCA



PCA cannot capture NON-LINEAR structure!

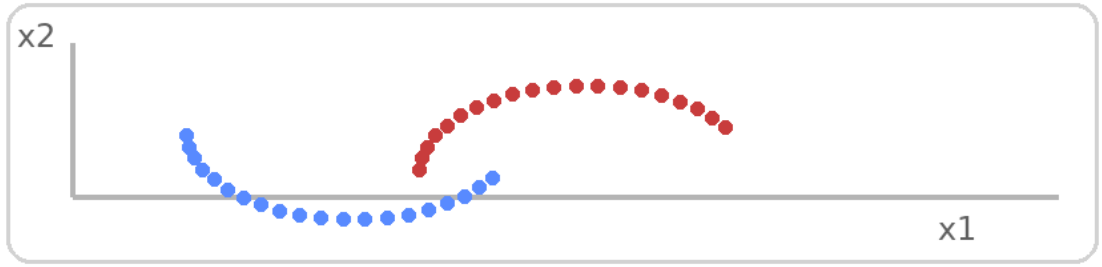
Myth Busters

MythBusters #3

“PCA can discover any low-dimensional structure.”

Myth: if the data is low-dimensional in some sense, PCA will always recover it.

Reality: PCA is a linear method. The lecture explicitly points out that PCA cannot capture nonlinear structure.



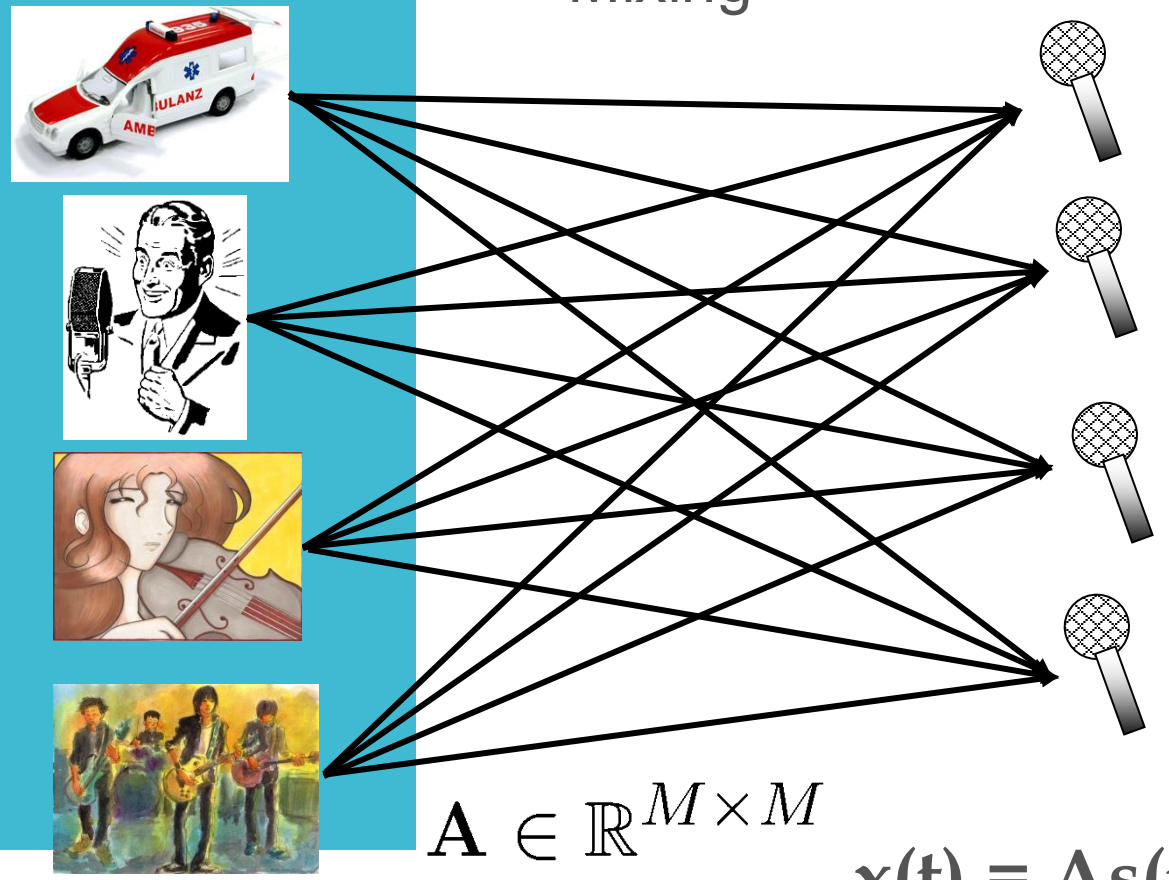
This is why nonlinear methods (e.g. autoencoders) can be appealing.

The Cocktail Party Problem

Sources

Mixing

Observation



$\mathbf{A} \in \mathbb{R}^{M \times M}$

$\mathbf{x(t) = As(t)}$

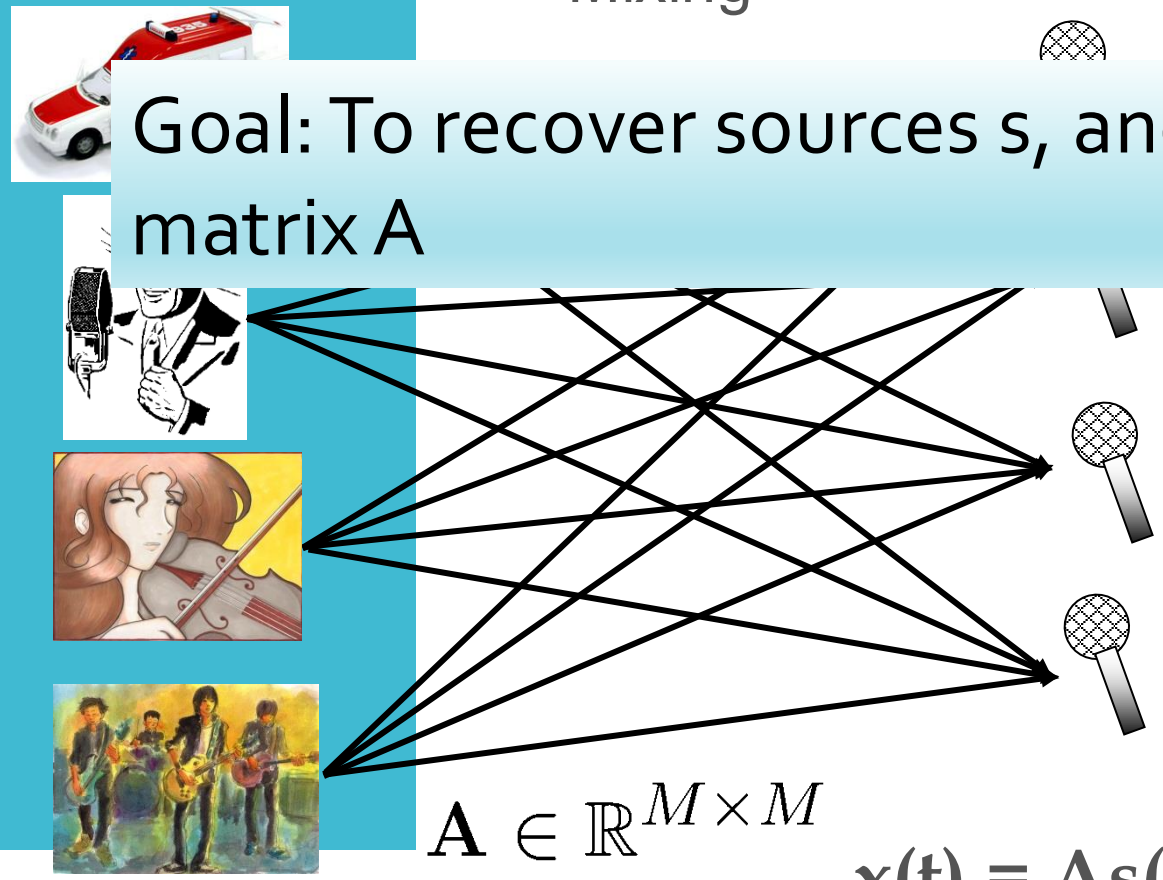
s(t)

The Cocktail Party Problem

Sources

Mixing

Observation



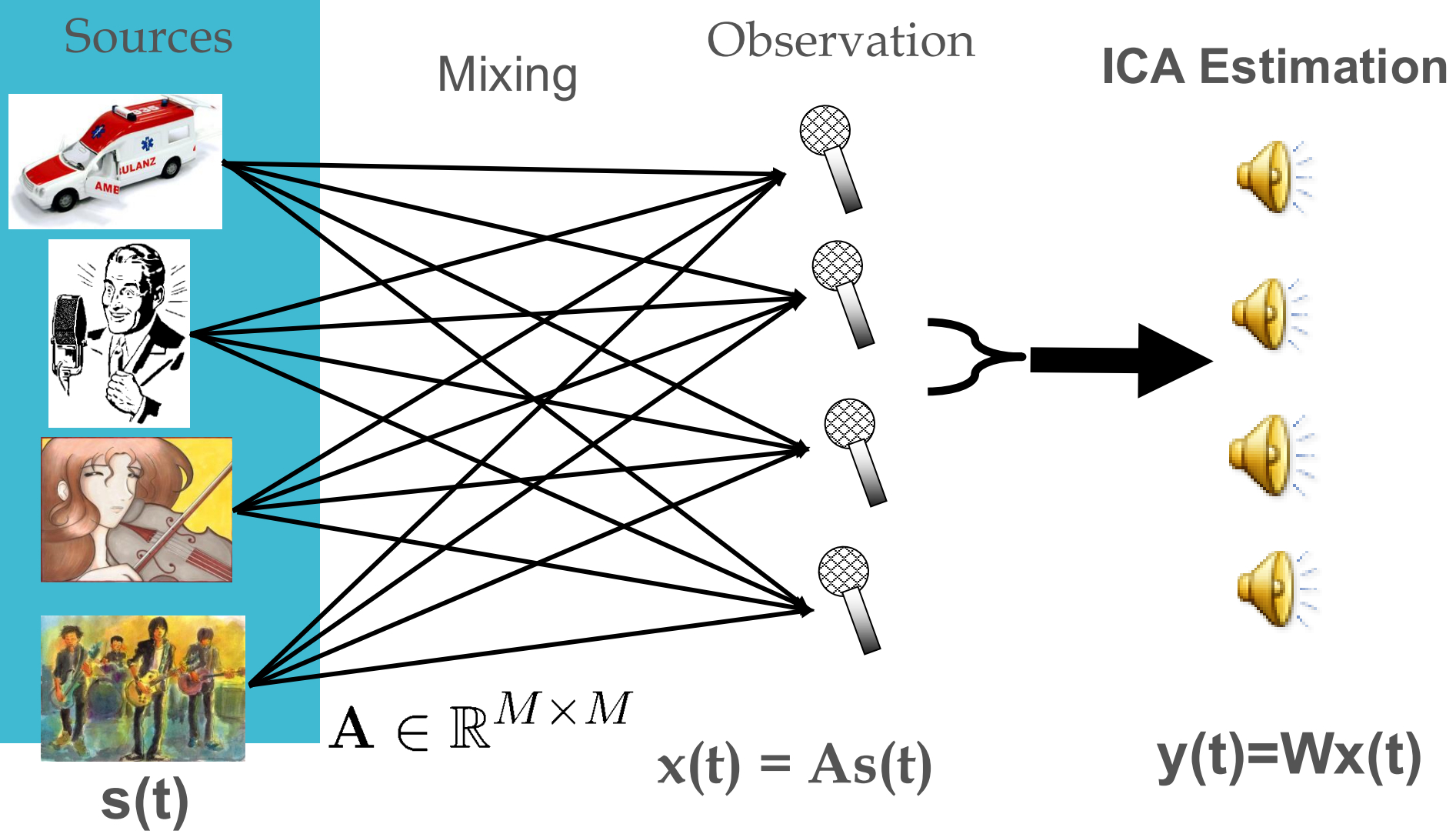
Goal: To recover sources s, and mixing matrix A

$\mathbf{A} \in \mathbb{R}^{M \times M}$

$\mathbf{x(t) = As(t)}$

s(t)

The Cocktail Party Problem
SOLVING WITH ICA



The Cocktail Party Problem
SOLVING WITH ICA

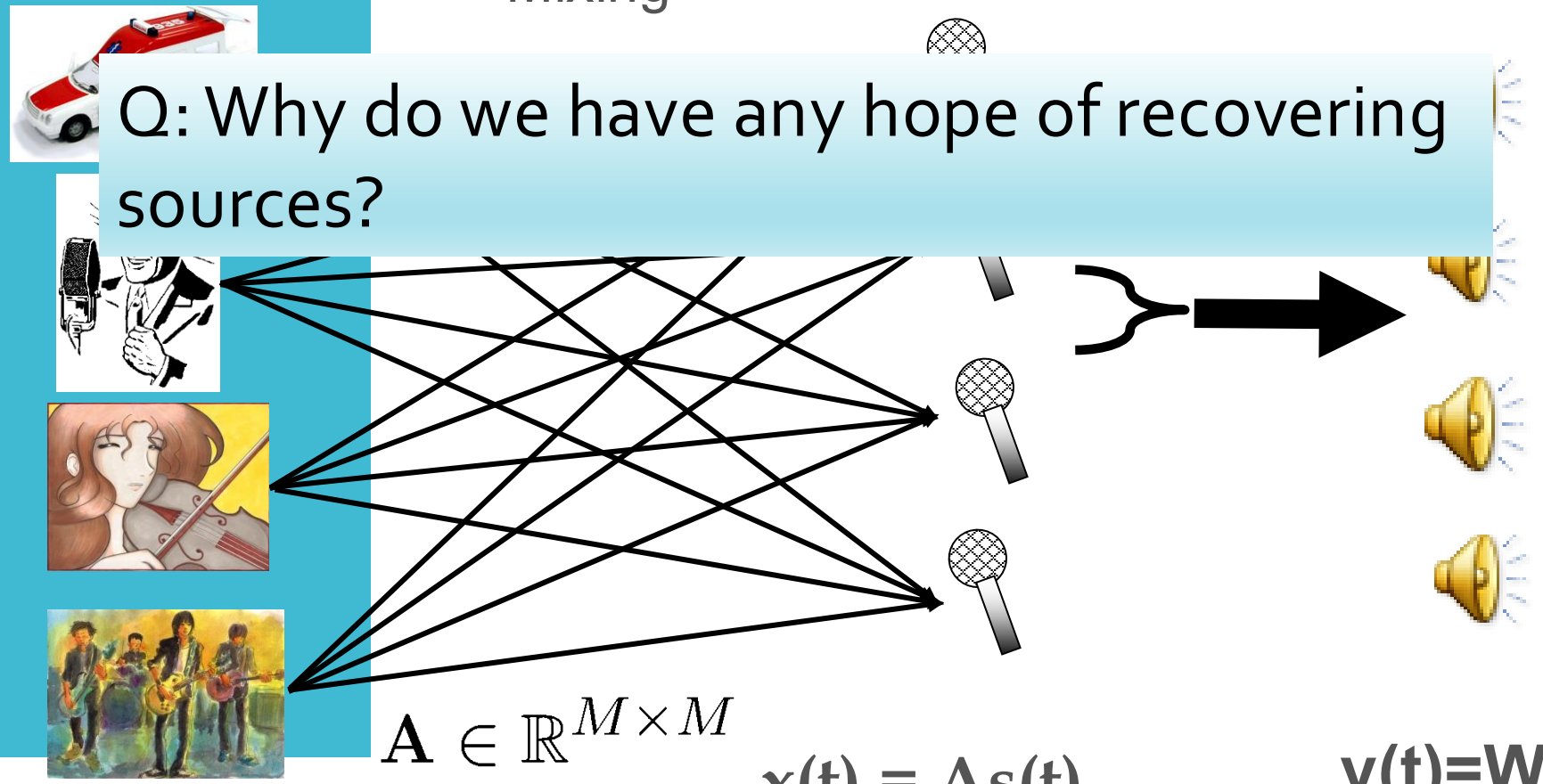
Sources

Mixing

Observation

ICA Estimation

Q: Why do we have any hope of recovering sources?



$\mathbf{A} \in \mathbb{R}^{M \times M}$

s(t)

x(t) = As(t)

y(t)=Wx(t)

The Cocktail Party Problem
SOLVING WITH ICA

Sources

Mixing

Observation

ICA Estimation

Q: Why do we have any hope of recovering sources?

A: Assume that sources are independent; find a linear transformation of data that is independent

$\mathbf{A} \in \mathbb{R}^{M \times M}$

$\mathbf{x(t) = As(t)}$

y(t)=Wx(t)

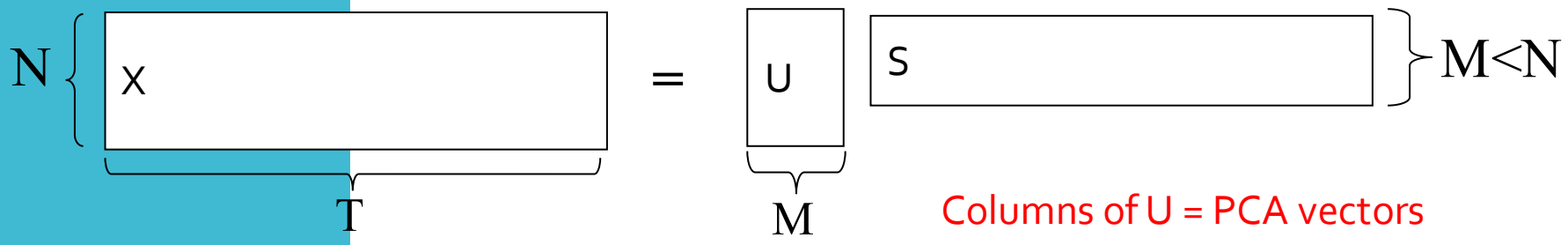
s(t)

ICA vs PCA

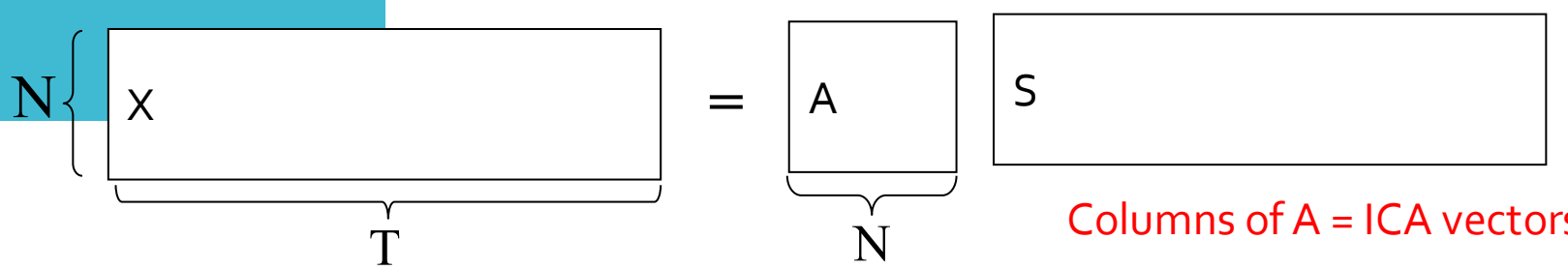
Perform linear transformations

Matrix factorization

PCA: ***low rank*** matrix factorization for ***compression***



ICA: ***full rank*** matrix factorization to ***remove dependency*** among the rows



ICA vs PCA

- PCA: **$X=US$, $U^TU=I$**
- ICA: **X=AS, A is invertible**

- PCA **does** compression
  - M<N

- ICA does **not** do compression
  - same # of features (M=N)

- PCA just removes correlations, **not** higher order dependence
- ICA removes correlations, **and** higher order dependence

- PCA: some components are **more important** than others (based on eigenvalues)
- ICA: components are **equally important**

Independent Component Analysis (ICA)

- Assume our data is a linear transformation of arbitrary (not necessarily orthogonal) components ("signals")

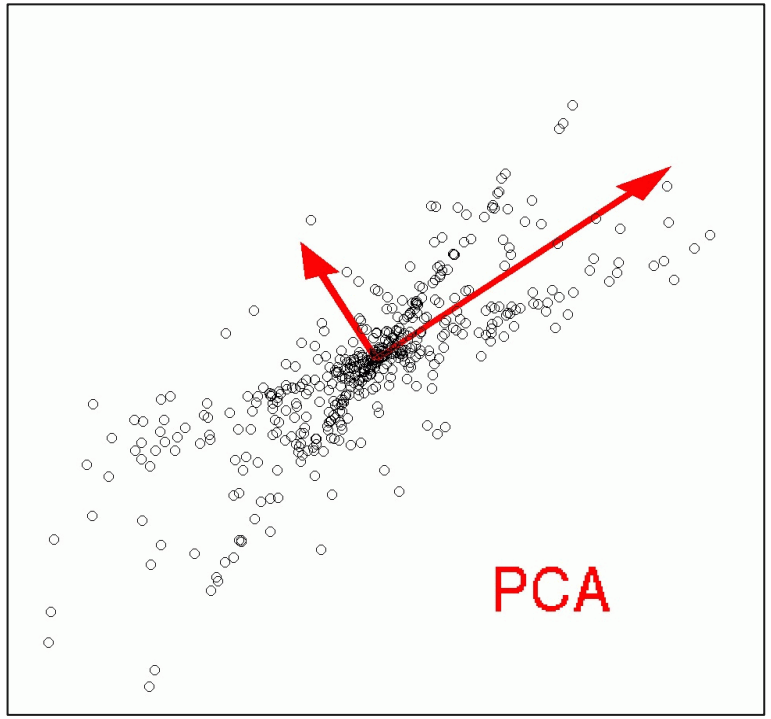
$$\boldsymbol{x}^{(i)} = A\boldsymbol{s}^{(i)}$$

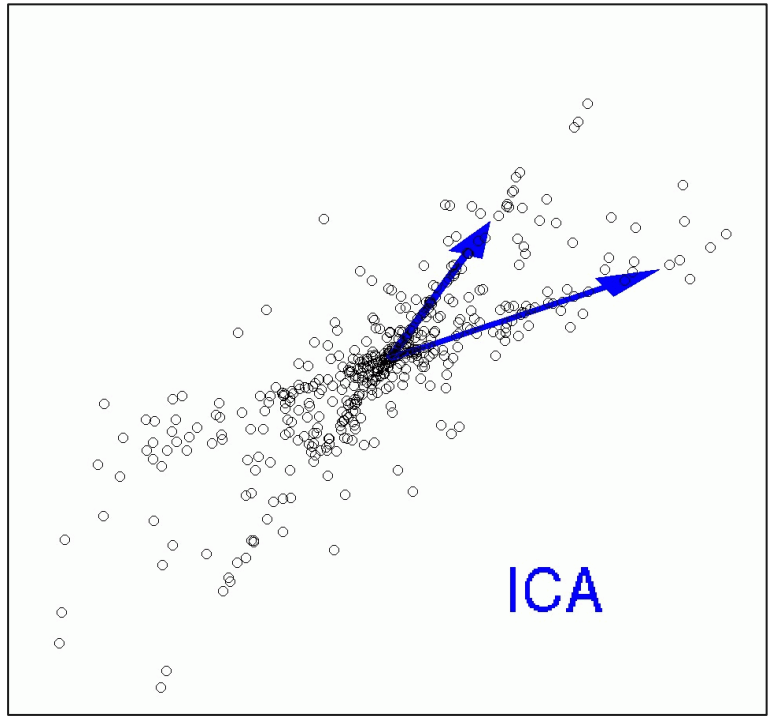
  - Typically assume $\boldsymbol{s}^{(i)}$ is the same size as $\boldsymbol{x}^{(i)}$

- Goal: find components that are as statistically independent as possible:

$$p\left(s_1^{(i)}, \dots, s_D^{(i)}\right) \approx p\left(s_1^{(i)}\right) \dots p\left(s_D^{(i)}\right)$$

- Common approach: minimize the mutual information between $s_1^{(i)}, \dots, s_D^{(i)}$

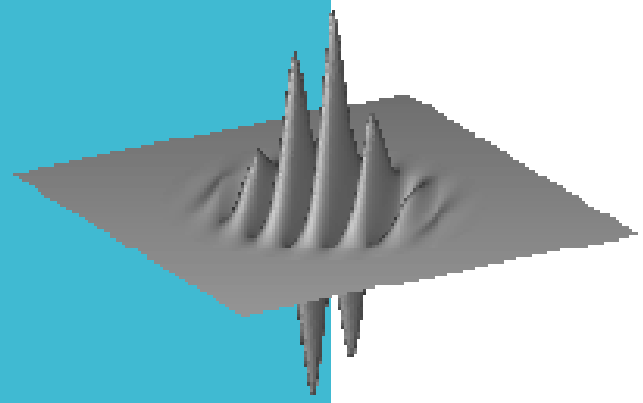


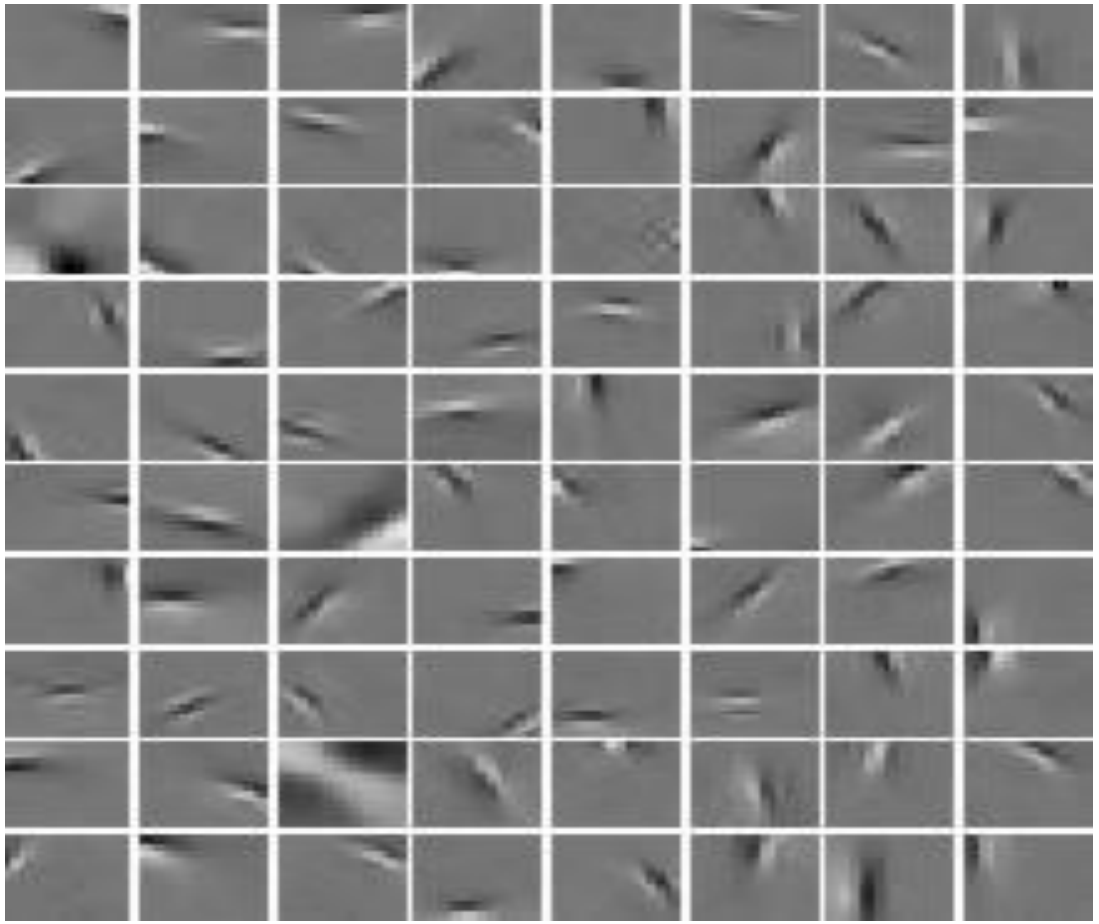


PCA vs. ICA

Figure courtesy of Barnabás Póczos

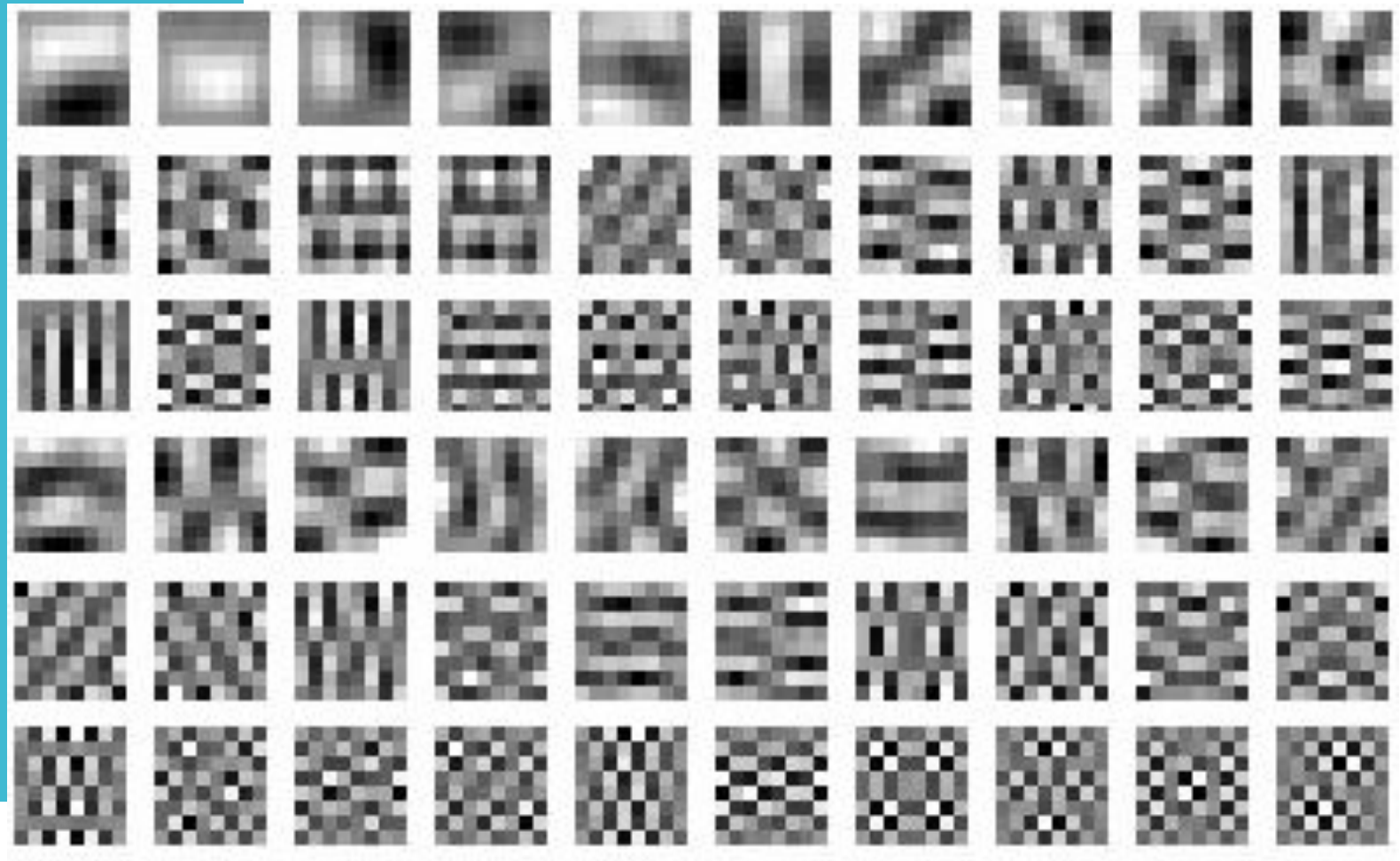
ICA basis vectors extracted from natural images





Gabor wavelets,
edge detection,
receptive fields of V1 cells..., deep neural networks

PCA basis vectors extracted from natural images



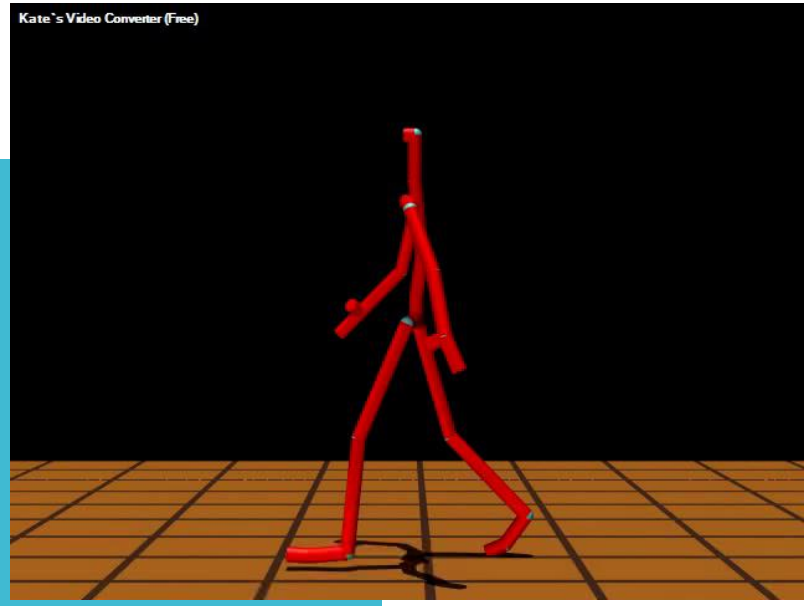
ICA for Motion Style Components

- Method for analysis and synthesis of human motion from motion captured data
- Provides perceptually meaningful “style” components
- 3D Motion capture - data matrix
- 109 markers (327 dim data)

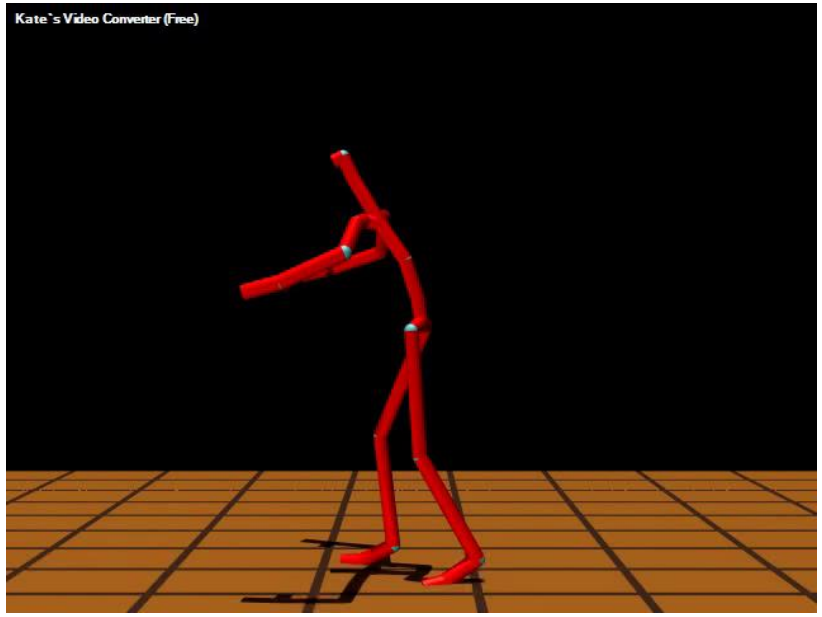
Goal: Find motion style components.

ICA - 6 independent components (emotion, content,…)

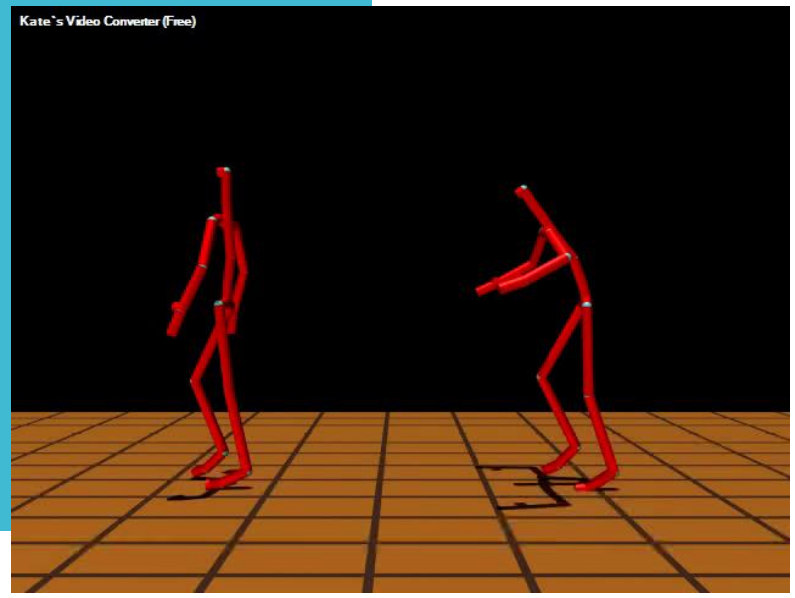
(Mori & Hoshino 2002, Shapiro et al 2006, Cao et al 2003)



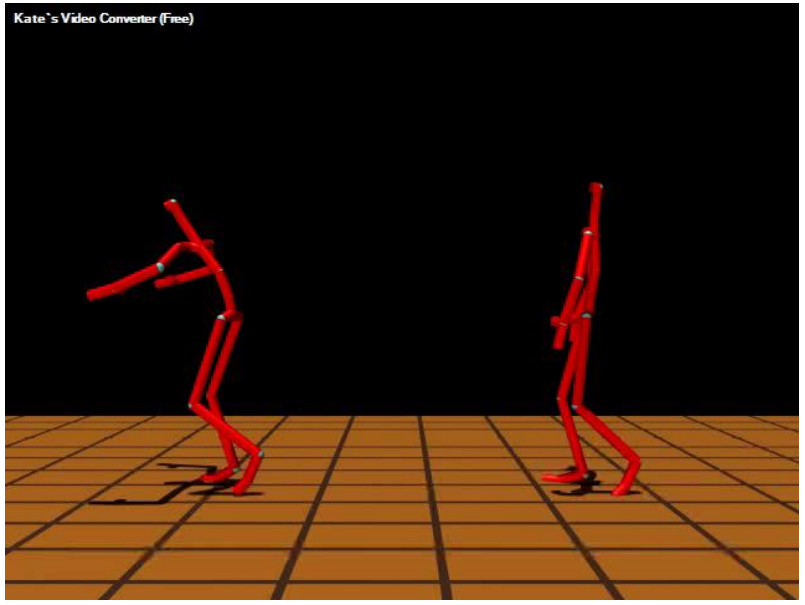
walk



sneaky



walk with sneaky



sneaky with walk

Quick Game

Two truths and a lie (dimensionality reductio

Vote for the lie: A, B, or C.

A. In PCA, the first principal component is the eigenvector corresponding to the largest eigenvalue.

B. ICA looks for statistically independent components, not necessarily orthogonal ones.

C. Standard ICA is mainly a compression method that keeps only a few components, just like PCA.

Quick Game

Two truths and a lie (dimensionality reductio

Vote for the lie: A, B, or C.

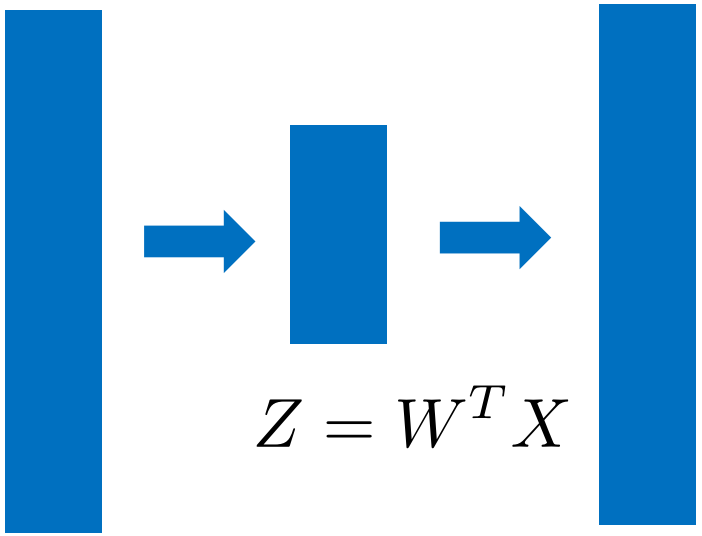
A. In PCA, the first principal component is the eigenvector corresponding to the largest eigenvalue.

B. ICA looks for statistically independent components, not necessarily orthogonal ones.

C. Standard ICA is mainly a compression method that keeps only a few components, just like PCA.

Reveal: C is the lie. The lecture says ICA typically keeps the same dimensionality; PCA is the compres

PCA as Linear Auto-encoder



$Z = W^T X$

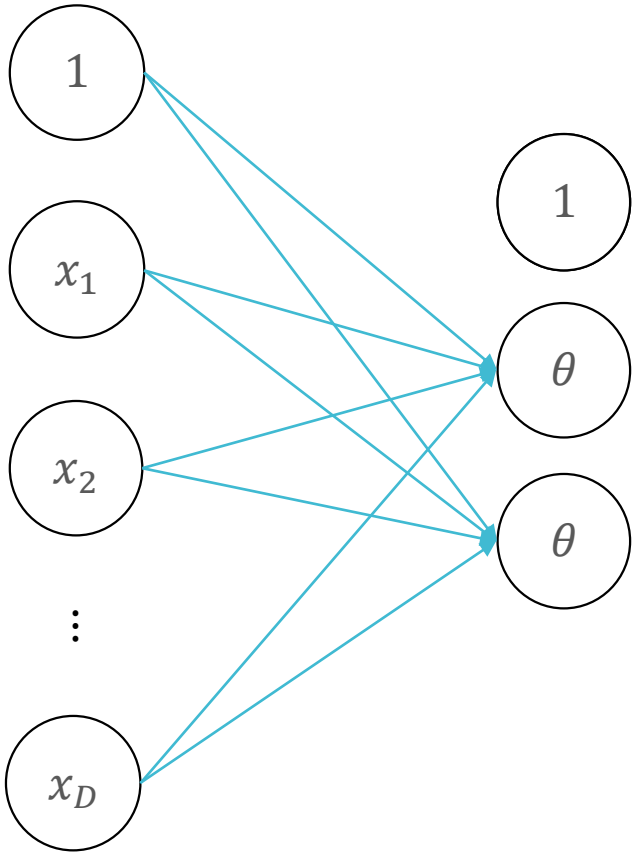
X

$\tilde{X} = WZ = WW^T X$

It can be shown that minimizing squared error as earlier is to minimizing reconstruction error

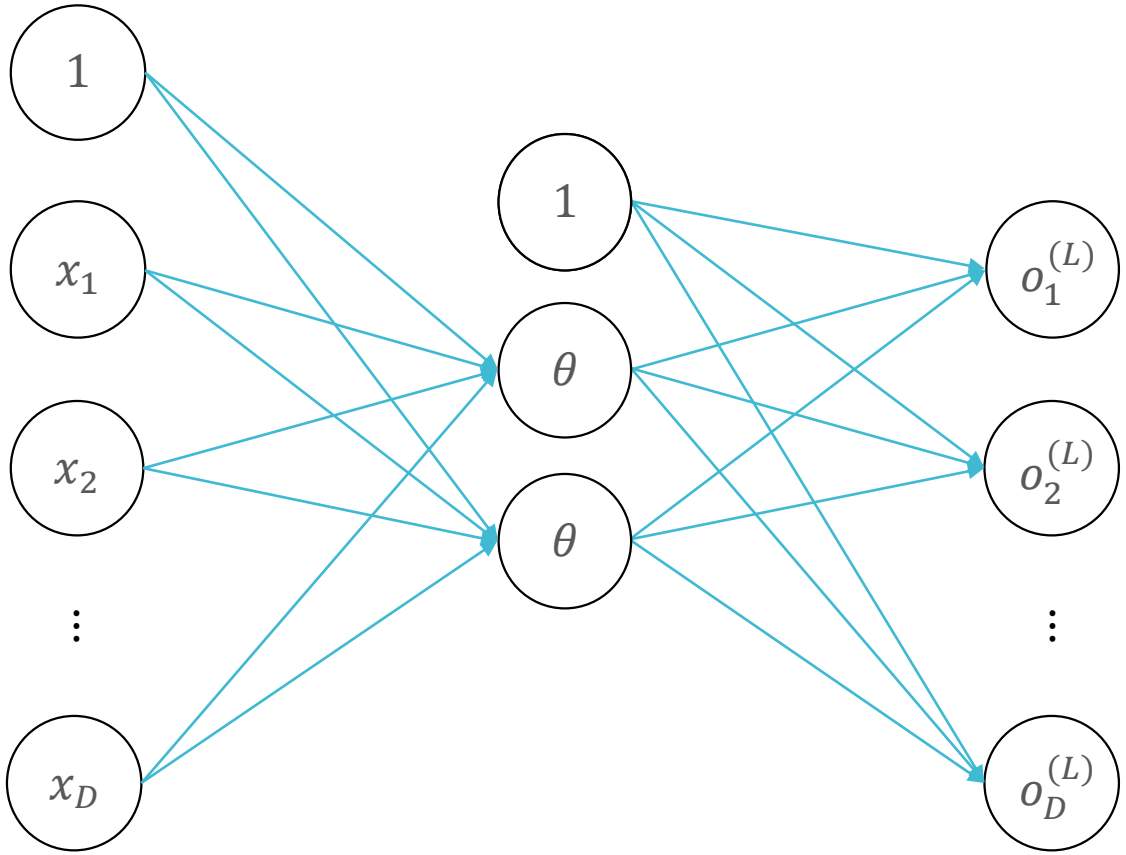
$$\|\tilde{X} - X\|_2^2 = \|WW^T X - X\|_2^2$$

Autoencoders



Insight: neural networks implicitly learn low-dimensional representations of inputs in hidden layers

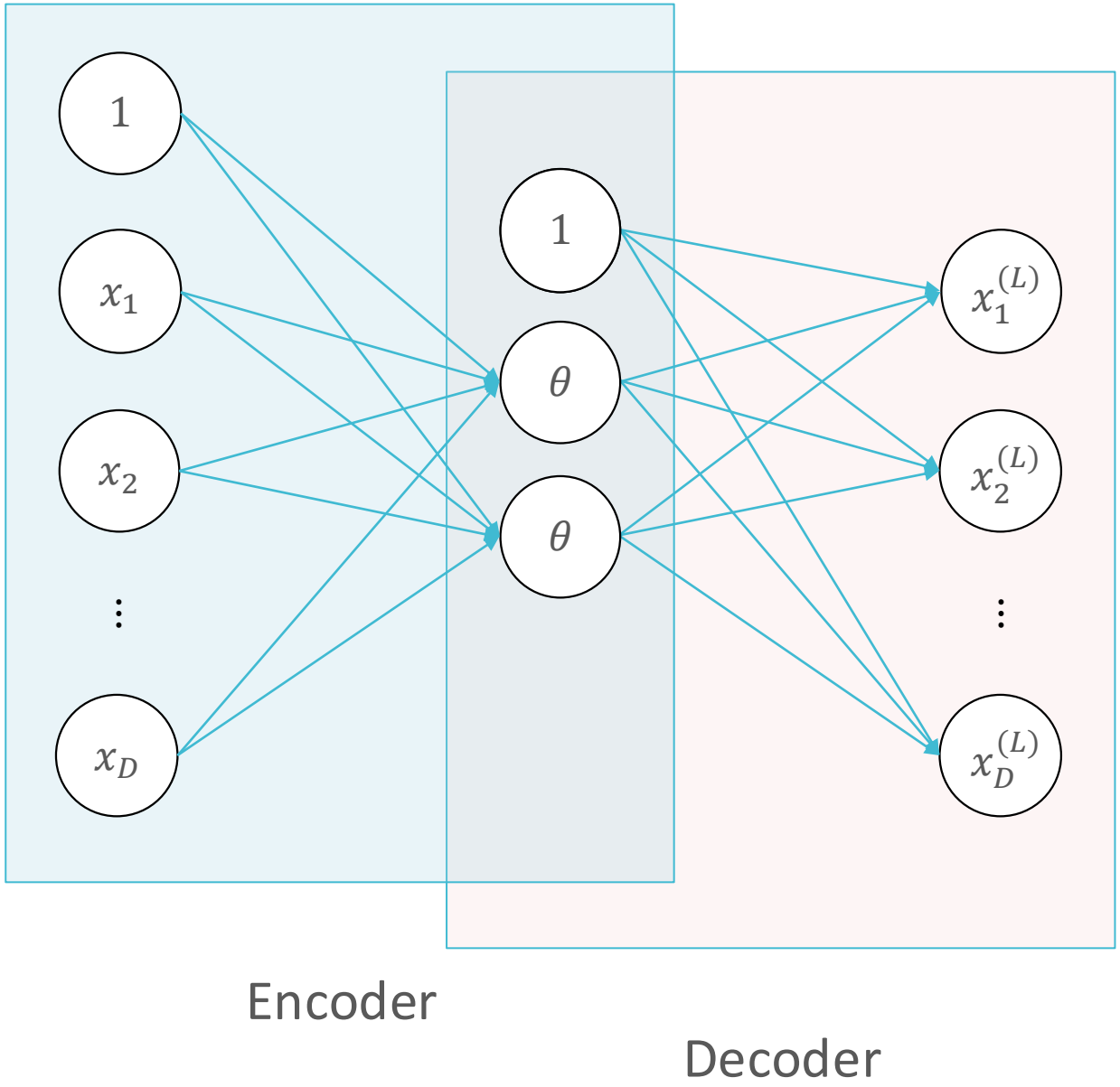
Autoencoders



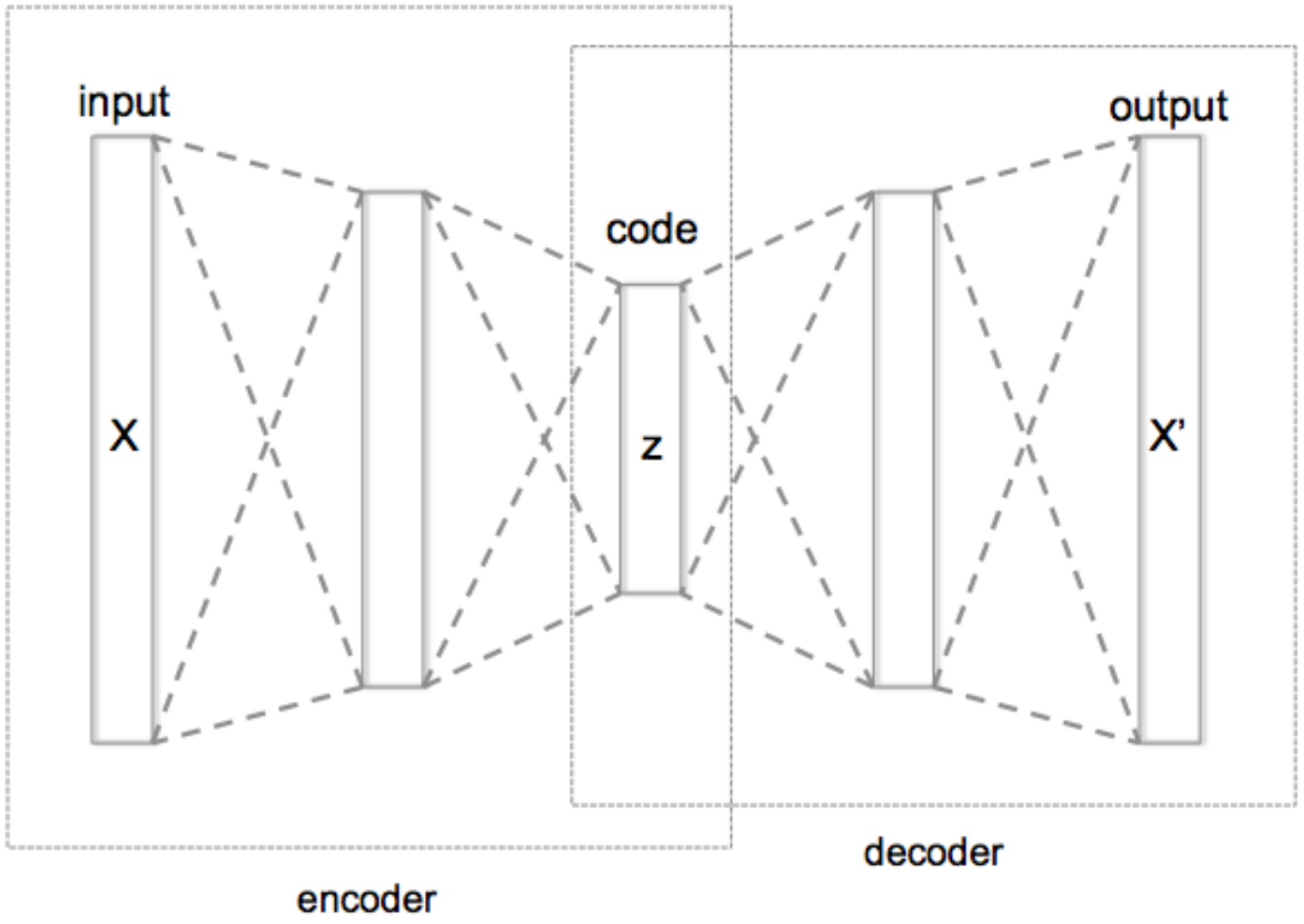
- Learn the weights by minimizing the reconstruction loss:

$$e(\boldsymbol{x}) = \left\| \boldsymbol{x} - \boldsymbol{o}^{(L)} \right\|_2^2$$

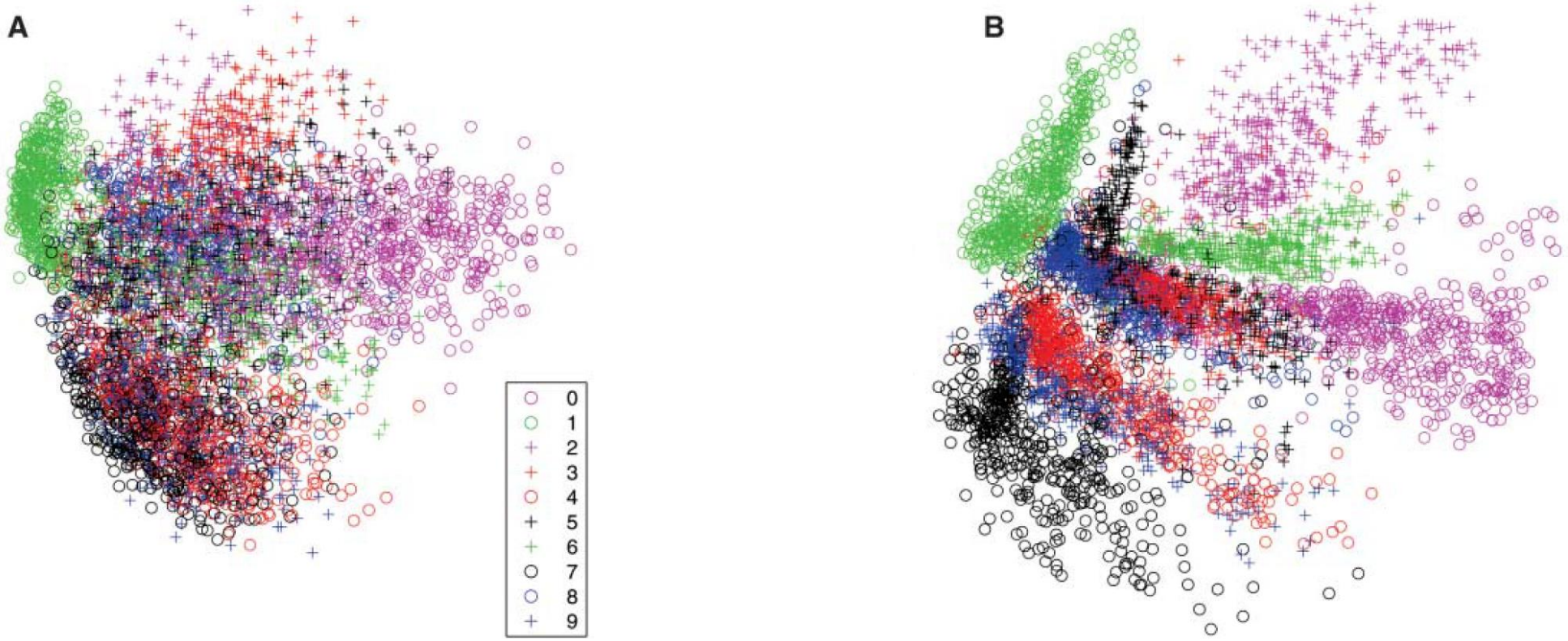
Autoencoders



Deep Autoencoders



Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png



PCA (A) vs. Autoencoders (B) (Hinton and Salakhutdinov, 2014)

Source: https://www.science.org/doi/10.1126/science.1127647

Data Representation

- The topmost realization over past decade: important to use a good **representation of** the data
- Example problem: Given a chair, predict if it is ergonomic
  - Features of chairs?



Data Representation

- It is common to represent data items as a vector of features
- What are these features?
  - organically specified: e.g. values of various blood tests given a blood sample data item, pixel values in a chair image
  - hand-constructed: number of legs in chair, …
- Approaches for dimensionality reduction can also be thought of as "representation learning" methods!

Neural Representation Learning

- Modern approaches extend beyond such (denoising) neural auto-encoders to:
  - Variational Auto-Encoders (VAEs)
  - Generative Adversarial Networks (GANs)
  - Diffusion Models
  - Contrastive Learning
  - …

Key Takeaways

- PCA finds an orthonormal basis where the first principal component maximizes the variance $\Leftrightarrow$ minimizes the reconstruction error
- ICA finds statistically independent, not orthogonal components
- Autoencoders use neural networks to automatically learn a latent representation that minimizes the reconstruction error

Final Poll

Poll Link:

Final poll (pick one)

Which statement is correct according to the lecture?

A. PCA removes all statistical dependence between features.

B. ICA seeks orthogonal directions of maximum variance.

C. Autoencoders must be linear, so they cannot represent nonlinear structure.

D. PCA seeks orthogonal directions that preserve variance / minimize reconstruction error, while ICA seeks statistically independent components.

E. PCA is the right tool whenever the data lies on a nonlinear manifold.