Signal Processing

Lecture 16: Dimensionality Reduction

Konstantinos Chatzilygeroudis - costashatz@upatras.gr

Department of Electrical and Computer Engineering
University of Patras

Template made by Panagiotis Papagiannopoulos

Why Dimensionality Reduction?

We often observe data in a *high-dimensional* space:

$$\boldsymbol{x}_i \in \mathbb{R}^D, \quad i = 1, \ldots, N,$$

where D can be large (pixels, sensors, features, embeddings, ...).

But the phenomenon that generated the data may be much simpler:

$$\boldsymbol{x}_i \approx g(\boldsymbol{z}_i), \qquad \boldsymbol{z}_i \in \mathbb{R}^d, \;\; d \ll D.$$

What are we trying to achieve?

We want a representation

$$\boldsymbol{z}_i = g(\boldsymbol{x}_i) \in \mathbb{R}^d \quad (d \ll D)$$

that preserves the *important* structure of the data.

Typical goals:

- **Compression:** store/transmit $\boldsymbol{x}_i$ using fewer numbers.
- **Visualization:** understand structure by plotting $d = 2$ or $d = 3$.
- **Denoising:** remove directions that are mostly noise.
- **Learning efficiency:** reduce computation and sample complexity.

The curse of dimensionality (intuition)

As D grows, geometric intuition changes:

- Data become *sparse*: a fixed number of samples N covers a vanishing fraction of $\mathbb{R}^D$.
- Distances can become less informative (many points look "equally far").
- Estimating quantities like densities or covariances becomes harder without much larger N.

In many problems, we pay a price in high dimensions:

more features $\Rightarrow$ more parameters to estimate $\Rightarrow$ more data needed.

Redundancy and correlations

High-dimensional features are often *correlated* or *redundant*.

Example idea:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{bmatrix} \quad \text{with} \quad x_2 \approx 2x_1, \;\; x_5 \approx x_1 + x_3, \;\; \ldots$$

So although $\boldsymbol{x} \in \mathbb{R}^D$, the data may concentrate near a much lower-dimensional set (e.g., a line, plane, or curved manifold).

A geometric picture

Even if observations live in $\mathbb{R}^D$, the data may lie (approximately) in a lower-dimensional subspace:

$$\boldsymbol{x}_i \approx \boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_i, \qquad \boldsymbol{W} \in \mathbb{R}^{D\times d}, \;\; \boldsymbol{z}_i \in \mathbb{R}^d, \;\; d \ll D.$$

Interpretation:

- $\boldsymbol{\mu}$ is the mean (translation).
- Columns of $\boldsymbol{W}$ span a d-dimensional subspace capturing most variation.
- Remaining directions contain little signal (often noise).

Formalizing "keep what matters"

A common principle: preserve information measured by *variance* (or energy).

If $\boldsymbol{P}$ is a projection onto a d-dimensional subspace, we want

$$\|\boldsymbol{x}_i - \boldsymbol{P}\boldsymbol{x}_i\|^2 \text{ small for most } i.$$

Equivalently, we want $\boldsymbol{P}\boldsymbol{x}_i$ to retain as much "spread" of the data as possible.

In other words, we want to:

- choose directions of **maximum variance**,
- or equivalently, minimize **reconstruction error**.

A symmetric 2×2 matrix as a geometric transform

Let

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \quad \text{(symmetric)}.$$

It defines a linear transformation $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$.

Key geometric fact (symmetric case):

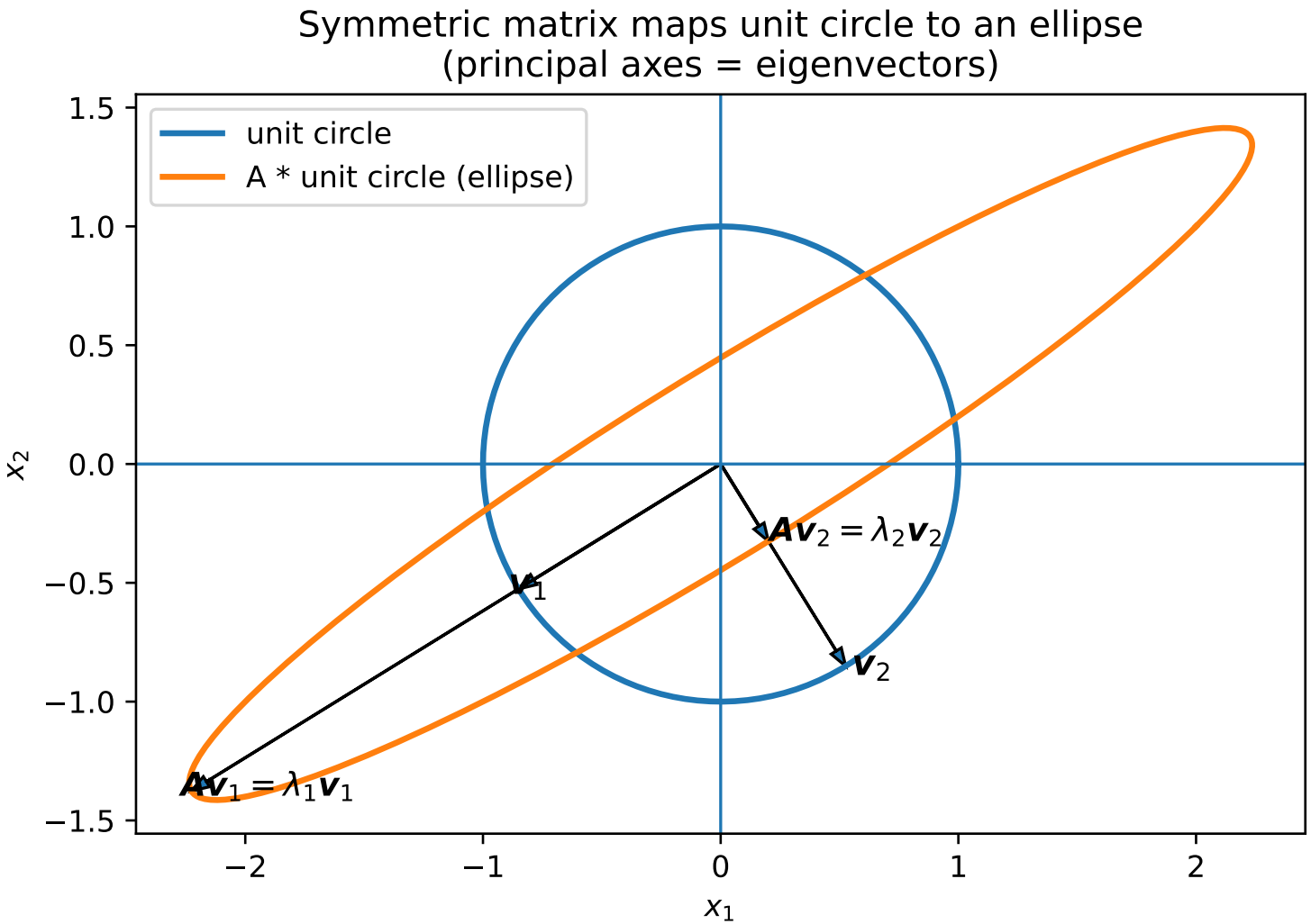
- $\boldsymbol{A}$ maps the **unit circle** to an **ellipse**.
- The ellipse's principal axes are **orthogonal**.
- Those principal axes directions are the **eigenvectors** of $\boldsymbol{A}$.

Eigenpairs satisfy:

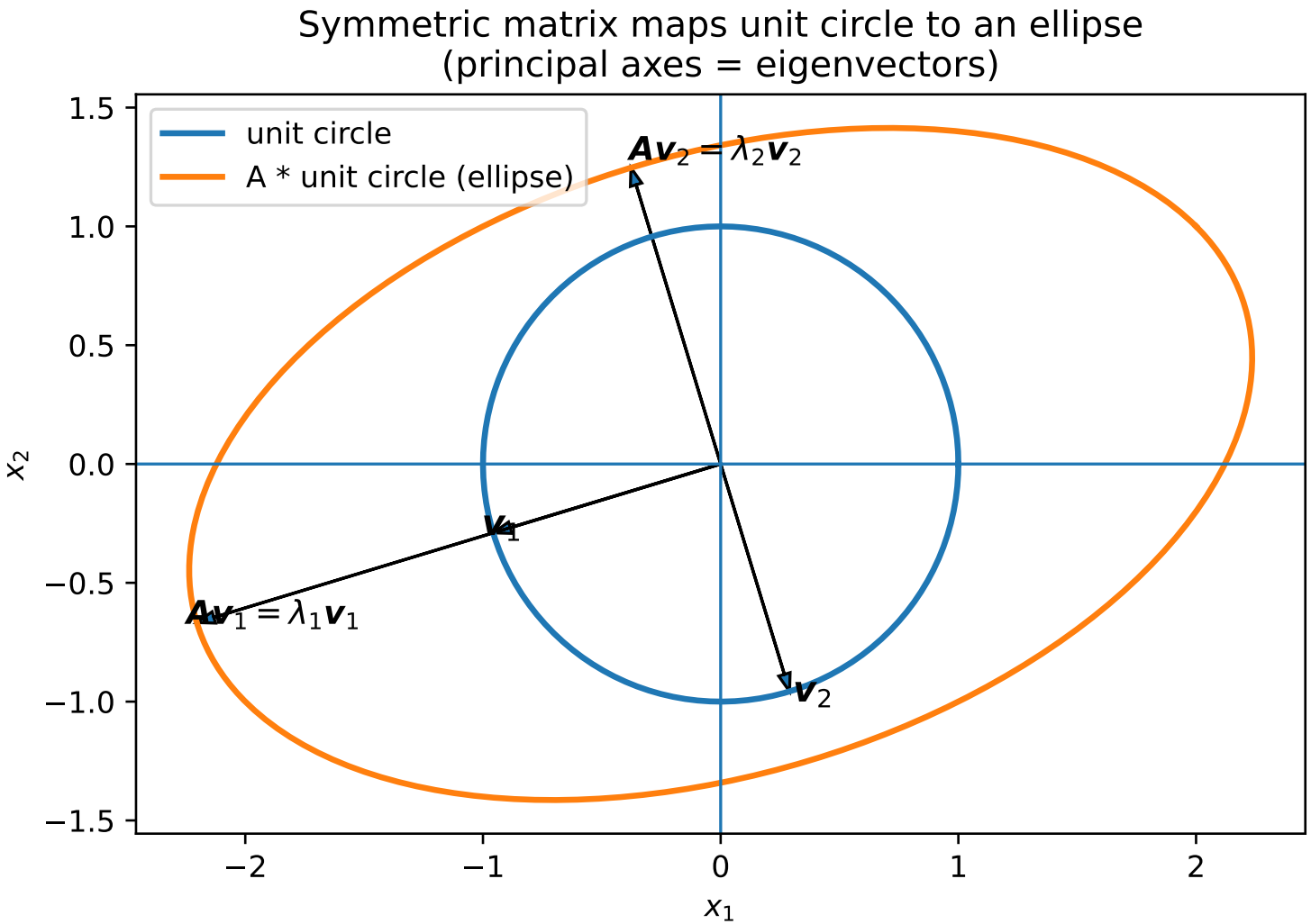
$$\boldsymbol{A}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k.$$

Along direction $\boldsymbol{v}_k$, vectors are scaled by λ_k (no rotation).

Visualizing the transformation



Visualizing the transformation (2)



Unit circle $\rightarrow$ ellipse (principal axes = eigenvectors)

Consider all unit vectors:

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^2 : \|\boldsymbol{x}\| = 1\}.$$

Their images under $\boldsymbol{A}$ form:

$$\mathcal{E} = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathcal{C}\}.$$

For symmetric $\boldsymbol{A}$, $\mathcal{E}$ is an ellipse.

If $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top$ with

$$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2), \quad \boldsymbol{Q} = [\boldsymbol{v}_1\ \boldsymbol{v}_2], \quad \boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I},$$

then:

- the ellipse axes point along $\boldsymbol{v}_1, \boldsymbol{v}_2$,
- the axis "length scales" are $|\lambda_1|$ and $|\lambda_2|$ (applied to unit vectors).

Rayleigh quotient: "which direction stretches the most?"

For any nonzero $\boldsymbol{x}$ define the Rayleigh quotient:

$$R(\boldsymbol{x}) = \frac{\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}{\boldsymbol{x}^\top \boldsymbol{x}}.$$

If $\|\boldsymbol{x}\| = 1$, then $R(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$.

For symmetric $\boldsymbol{A}$:

$$\max_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = \lambda_{\max} \quad \text{and is achieved at } \boldsymbol{x} = \boldsymbol{v}_{\max}.$$

$$\min_{\|\boldsymbol{x}\|=1} \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} = \lambda_{\min} \quad \text{and is achieved at } \boldsymbol{x} = \boldsymbol{v}_{\min}.$$

What changes for a non-symmetric matrix?

Let

$$\boldsymbol{A} \in \mathbb{R}^{2\times 2}, \qquad \boldsymbol{A} \neq \boldsymbol{A}^{\top}.$$

Then $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$ can include **shear** and **streching-like** effects.

Key differences from the symmetric case:

- Eigenvalues may be **complex** (no real invariant directions).
- Real eigenvectors (if they exist) need **not be orthogonal**.
- The image of the unit circle is still an ellipse, but its principal axes are **not** generally given by eigenvectors.

Singular Value Decomposition (SVD)

For any matrix

$$\boldsymbol{A} \in \mathbb{R}^{m \times n},$$

there exists a decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top},$$

where:

- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ has orthonormal columns,
- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ has orthonormal columns,
- $\boldsymbol{\Sigma} \in \mathbb{R}^{m \times n}$ is diagonal with

$$\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant 0.$$

Geometric interpretation of $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$:

$$\boldsymbol{x} \xrightarrow{\boldsymbol{V}^{\top}} \text{rotate/reflect} \xrightarrow{\boldsymbol{\Sigma}} \text{axis-aligned scaling} \xrightarrow{\boldsymbol{U}} \text{rotate/reflect}.$$

Unit circle $\longrightarrow$ ellipse via SVD

Consider the unit circle $\{\boldsymbol{x} : \|\boldsymbol{x}\| = 1\}$ in $\mathbb{R}^2$.

Under $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$:

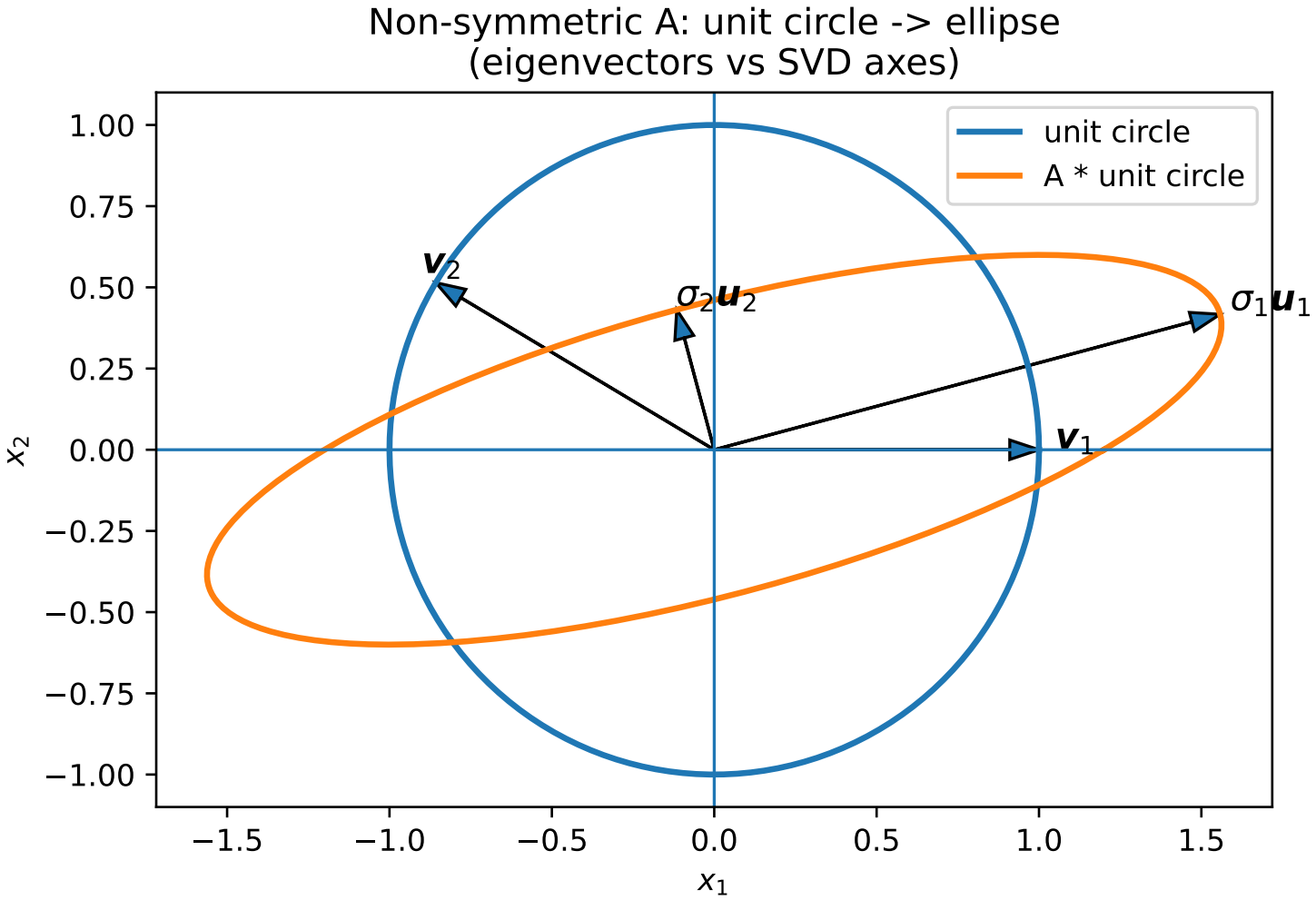
- columns of $\boldsymbol{V}$ give **input directions** that map to the ellipse axes,
- columns of $\boldsymbol{U}$ give the **output directions** of those axes,
- singular values σ_k are the **lengths** of the ellipse semi-axes.

Unlike eigenvectors:

- SVD always exists (for any matrix),
- directions are always orthogonal,
- geometry is always real and well-defined.

Key takeaway: eigenvectors explain symmetric matrices well; SVD explains *all* linear maps.

Visualizing SVD



Principal Component Analysis (PCA)

Given centered data $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}$ and covariance

$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top,$$

PCA finds an orthonormal basis $\boldsymbol{W}_d = [\boldsymbol{w}_1 \ \cdots \ \boldsymbol{w}_d]$ that best captures the data.

Variance-maximizing view:

$$\boldsymbol{w}_1 = \arg \max_{\|\boldsymbol{w}\|=1} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}, \qquad \boldsymbol{w}_k = \arg \max_{\|\boldsymbol{w}\|=1,\ \boldsymbol{w} \perp \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{k-1}} \boldsymbol{w}^\top \boldsymbol{\Sigma} \boldsymbol{w}.$$

Equivalent reconstruction view:

$$\boldsymbol{W}_d = \arg \min_{\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}} \sum_{i=1}^{N} \left\| \tilde{\boldsymbol{x}}_i - \boldsymbol{W} \boldsymbol{W}^\top \tilde{\boldsymbol{x}}_i \right\|^2.$$

Solution: $\boldsymbol{w}_k$ are the eigenvectors of $\boldsymbol{\Sigma}$, with eigenvalues

$$\boldsymbol{\Sigma} \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k, \qquad \lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant 0,$$

and the reduced coordinates are

$$\boldsymbol{z}_i = \boldsymbol{W}_d^\top \tilde{\boldsymbol{x}}_i.$$

PCA Example: 3D points on a plane

We observe points in 3D:

$$\boldsymbol{x}_i \in \mathbb{R}^3, \quad i = 1, \ldots, N.$$

But the data are *approximately planar* (intrinsic dimension ≈ 2):

$$\boldsymbol{x}_i \approx \boldsymbol{\mu} + u_i \, \boldsymbol{p}_1 + v_i \, \boldsymbol{p}_2 + \varepsilon_i \, \boldsymbol{n},$$

where $\boldsymbol{p}_1, \boldsymbol{p}_2$ span an (unknown) plane, $\boldsymbol{n}$ is its normal, and $|\varepsilon_i|$ is small.

Within that plane, the points form an "S" shape (nonlinear structure), so:

- dimension $3 \to 2$ should work well,
- dimension $3 \to 1$ will generally *lose* structure.

PCA goal (variance-maximizing view)

First, center the data:

$$\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i, \qquad \tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}.$$

Define the sample covariance:

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} \tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^{\top} \in \mathbb{R}^{3\times 3}.$$

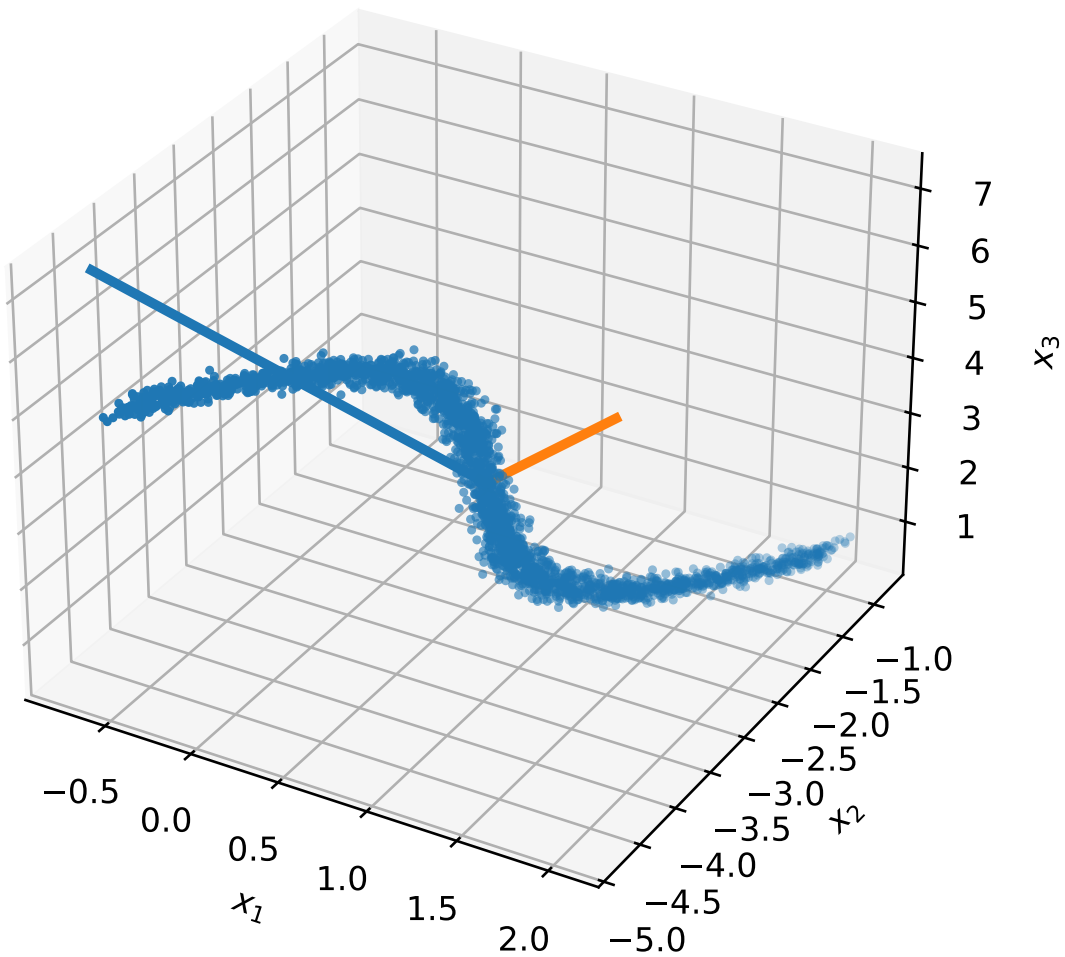
The first principal component solves

$$\max_{\|\boldsymbol{w}\|=1} \ \mathrm{Var}(\boldsymbol{w}^{\top}\tilde{\boldsymbol{x}}) \ = \ \max_{\|\boldsymbol{w}\|=1} \ \boldsymbol{w}^{\top}\boldsymbol{\Sigma}\,\boldsymbol{w}.$$

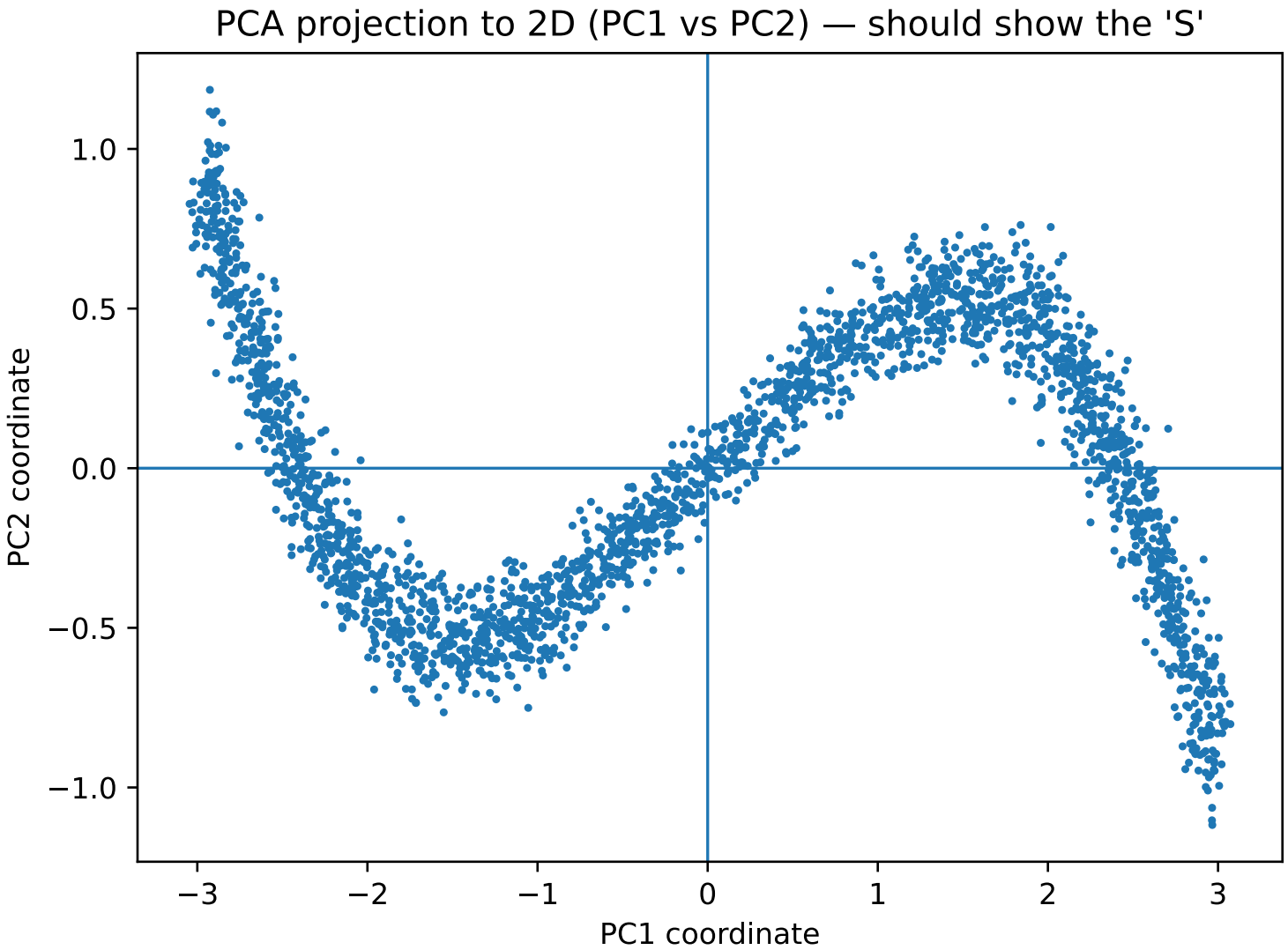
Thus $\boldsymbol{w}_1$ is the top eigenvector of $\boldsymbol{\Sigma}$, and the variance along it is λ_1.

Visualizing PCA

3D data: 'S' on a rotated plane + PCA axes



Visualizing PCA (2)



Why PCA discovers the plane (in our example)

Because the points are (approximately) on a plane:

$$\lambda_1 \text{ large}, \quad \lambda_2 \text{ large}, \quad \lambda_3 \text{ small}.$$

Geometrically:

- $\boldsymbol{w}_1, \boldsymbol{w}_2$ span the best-fitting plane (maximum retained variance),
- $\boldsymbol{w}_3$ approximates the plane normal (minimum variance direction).

We reduce dimension by projection:

$$\boldsymbol{W}_2 = [\boldsymbol{w}_1 \ \boldsymbol{w}_2] \in \mathbb{R}^{3\times 2}, \qquad \boldsymbol{z}_i = \boldsymbol{W}_2^\top \tilde{\boldsymbol{x}}_i \in \mathbb{R}^2.$$

The 2D coordinates $\boldsymbol{z}_i$ should reveal the "S" clearly.

Reconstruction and "information loss"

From reduced coordinates:

$$\hat{\boldsymbol{x}}_i = \bar{\boldsymbol{x}} + \boldsymbol{W}_2 \boldsymbol{z}_i$$

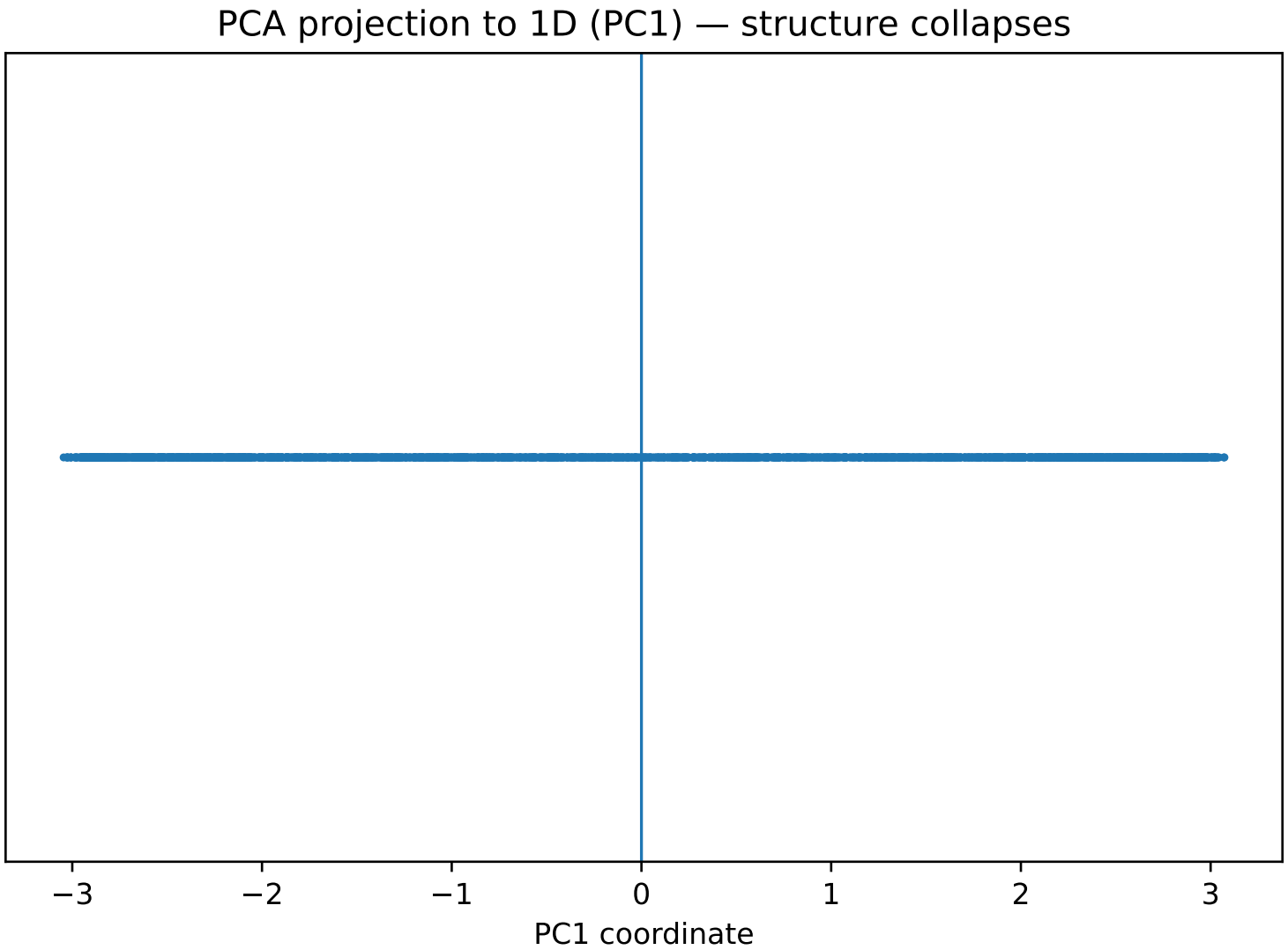
is the orthogonal projection of $\boldsymbol{x}_i$ onto the PCA plane.

Reconstruction error (per point):

$$\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 = \|\tilde{\boldsymbol{x}}_i - \boldsymbol{W}_2 \boldsymbol{W}_2^\top \tilde{\boldsymbol{x}}_i\|^2.$$

If we keep only 1 component ($\boldsymbol{W}_1 = \boldsymbol{w}_1$), the 1D projection can *collapse* parts of the "S" onto each other (structure loss).

Visualizing PCA (3)



PCA via SVD (same geometry, two viewpoints)

Stack centered samples as rows of a data matrix:

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1^\top \\ \tilde{\boldsymbol{x}}_2^\top \\ \vdots \\ \tilde{\boldsymbol{x}}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times D}, \qquad \tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}}.$$

Compute the SVD:

$$\tilde{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \qquad \boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \dots).$$

Then the sample covariance is

$$\boldsymbol{\Sigma}_{\text{data}} = \frac{1}{N}\tilde{\boldsymbol{X}}^\top \tilde{\boldsymbol{X}} = \boldsymbol{V}\left(\frac{\boldsymbol{\Sigma}^2}{N}\right)\boldsymbol{V}^\top.$$

So:

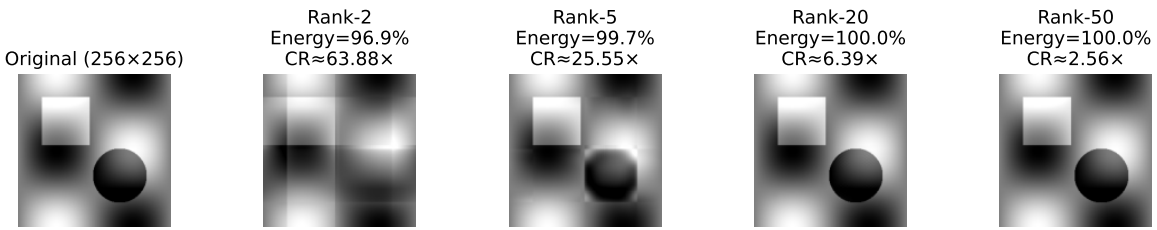
- **PCA directions** (principal axes) are columns of $\boldsymbol{V}$.
- **PCA eigenvalues** are $\lambda_k = \sigma_k^2/N$.
- **Reduced coordinates (scores)**:

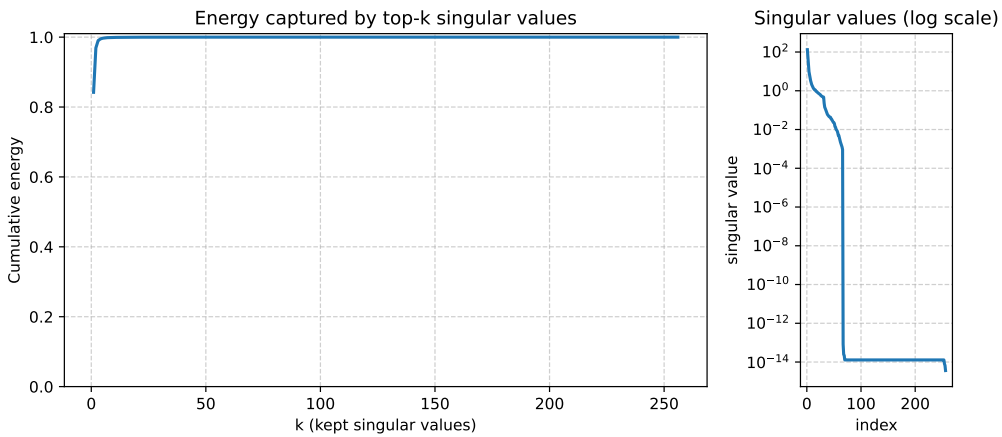
$$\boldsymbol{z}_i = \boldsymbol{V}_d^\top \tilde{\boldsymbol{x}}_i \quad \Leftrightarrow \quad \tilde{\boldsymbol{X}}\boldsymbol{V}_d = \boldsymbol{U}_d\boldsymbol{\Sigma}_d.$$

SVD for Image Compression (Grayscale)

- Treat an image as a matrix $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ (pixel intensities).
- Compute SVD: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$.
- Keep top-k singular values/vectors for compression:

$$\boldsymbol{X}_k = \boldsymbol{U}_{[:,1:k]}\, \boldsymbol{\Sigma}_{1:k,1:k}\, \boldsymbol{V}^{\top}_{[:,1:k]}.$$





Thank you

- **Any Questions?**
- **Office Hours**:
  - **Tue & Thu** *(09:00-11:00)*
  - 24/7 by email (`costashatz@upatras.gr`, subject: *ECE_SP_AM*)
- **Material and Announcements**

Laboratory of Automation & Robotics