Dimensionality reduction

Slides based on slides by Evimaria Terzi, Barnabás Póczos

Datasets in the form of matrices

We are given **n** objects and **d** features describing the objects.
(Each object has **d** numeric values describing it.)

Dataset

An **n-by-d** matrix **A**, A_{ij} shows the "***importance***" of feature **j** for object **i**.

Every row of **A** represents an object.

Goal

1. **Understand** the structure of the data, e.g., the underlying process generating the data.
2. **Learn important patterns**

Market basket matrices

d products
(e.g., milk, bread, wine, etc.)

n customers

$$A$$

A_{ij} = quantity of **j**-th product purchased by the **i**-th customer

Find a subset of the products that characterize customer behavior

Social-network matrices

d groups
(e.g., BU group, opera, etc.)

n users

$$A$$

A_{ij} = partitipation of the **i**-th user in the **j**-th group

Find a subset of the groups that accurately clusters social-network users

Document matrices

d terms
(e.g., theorem, proof, etc.)

n documents

$$A$$

A_{ij} = frequency of the **j**-th term in the **i**-th document

Find a subset of the terms that accurately clusters the documents

Recommendation systems

d products

n customers

$$A$$

A_{ij} = frequency of the **j**-th product is bought by the **i**-th customer

Find a subset of the products that accurately describe the behavior or the customers

Dimensionality reduction

- Dataset **X** consisting of **n** points in a **d**-dimensional space
- Data point $x_i \epsilon R^d$ (**d**-dimensional real vector):

  $x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$

- **Goal: Compress data into n points in R^K where K << d**
  - Retain as much information about the original data set
  - Retain desired properties of the original data set

Why Dimensionality Reduction?

- Reduce the size of the input (by reducing the dimensionality d to K)
  - You can learn faster/better a mixture of Gaussians
- Reduce the complexity and redundant information/noise
- Data compression
- Data Visualization
- …

Desired properties

- Original data can be reconstructed as good as possible
- Preserve the “relevant information” for our data mining task
- Preserve pairwise distances
- Reduce noise

Data Visualization

- Matrix format (65x53)

	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

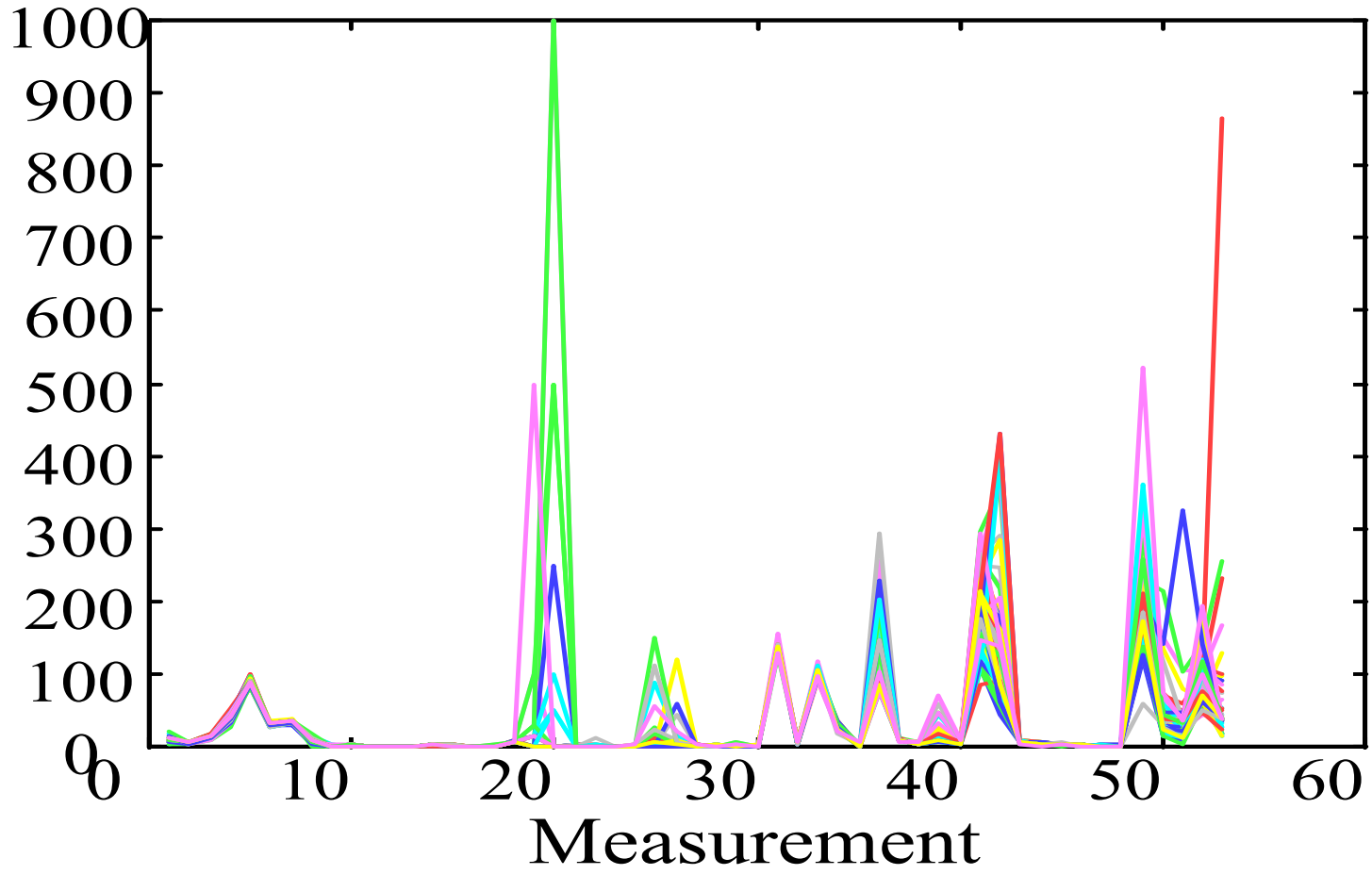
Instances

Features

Difficult to see the correlations between the features...

Data Visualization

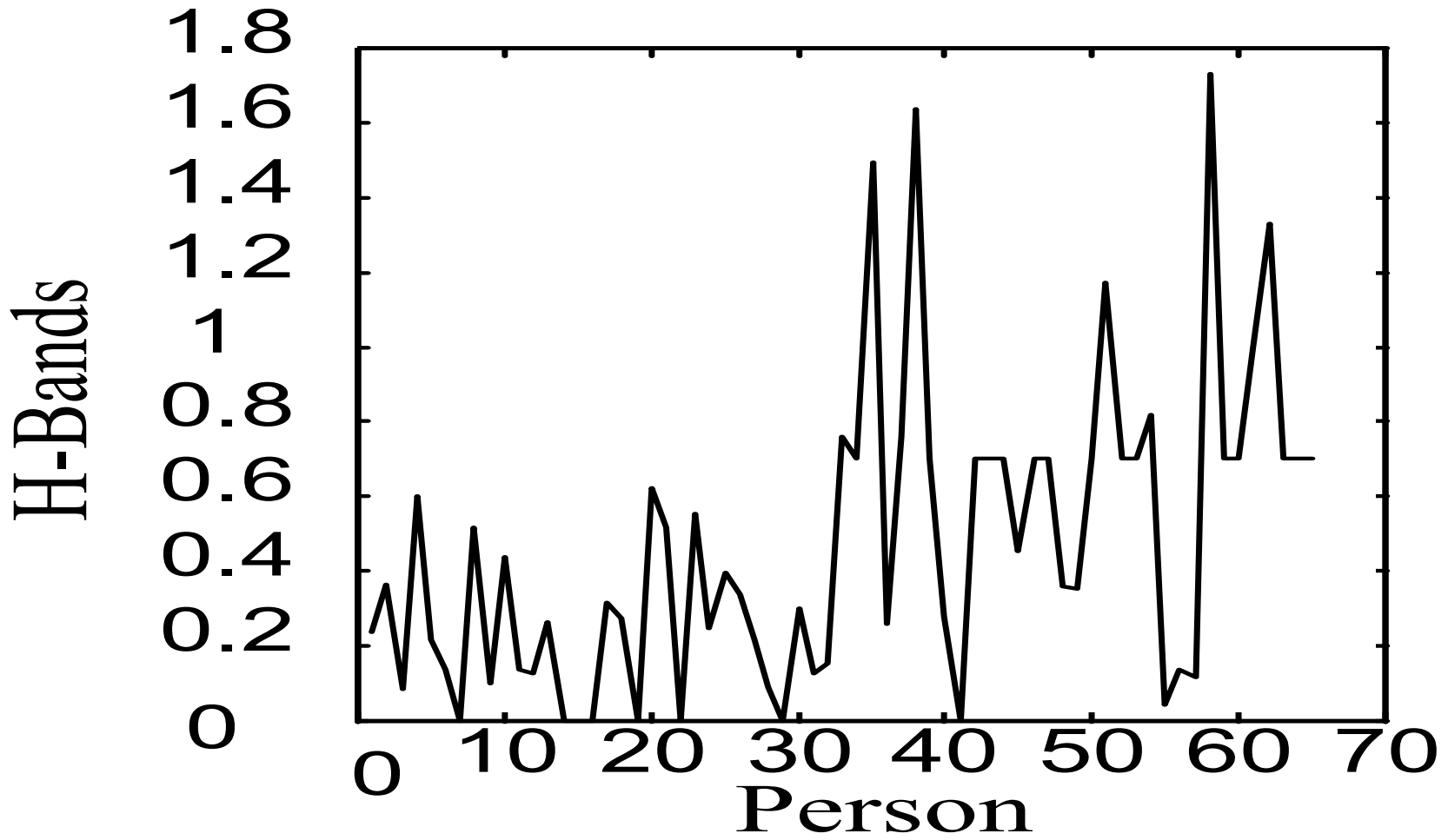
- Spectral format (65 pictures, one for each person)



Difficult to compare the different patients...

Data Visualization

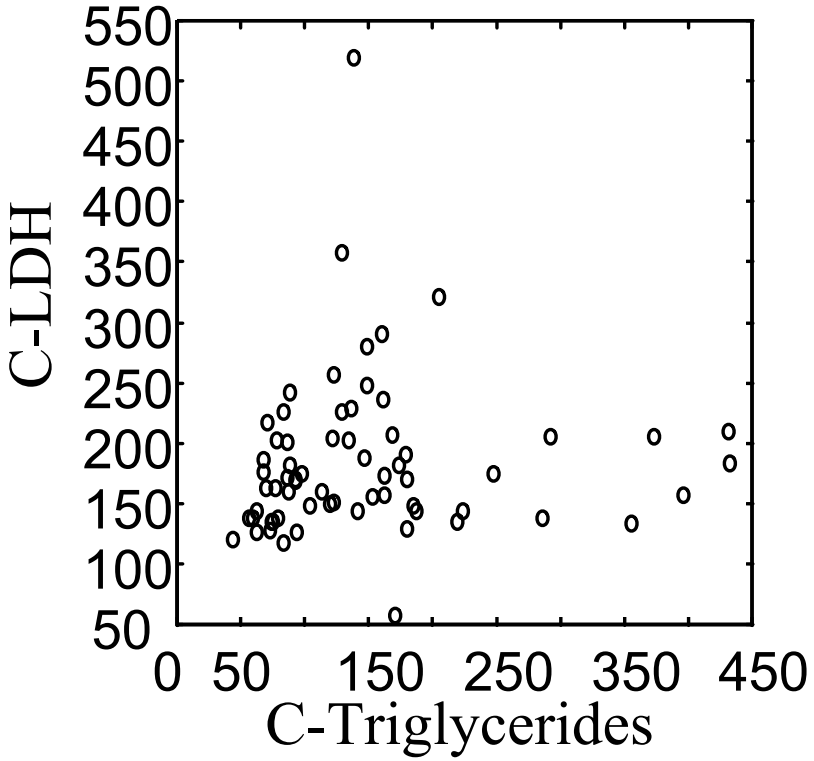
- Spectral format (53 pictures, one for each feature)



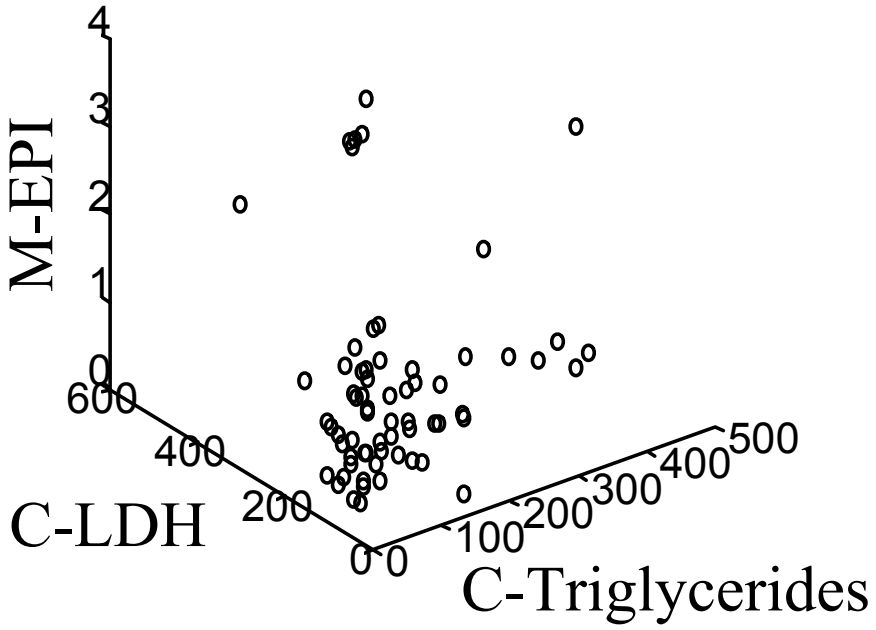
Difficult to see the correlations between the features...

Data Visualization

Bi-variate



Tri-variate



How can we visualize the other variables???

… difficult to see in 4 or higher dimensional spaces...

Data Visualization

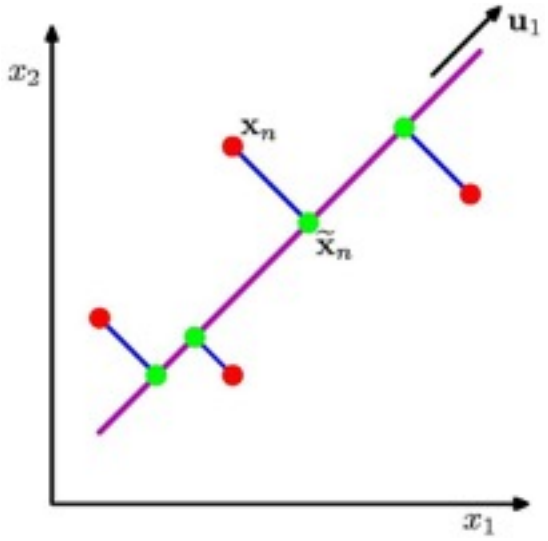
- Is there a representation better than the coordinate axes?

- 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?

- A solution: **Principal Component Analysis**

Principle Component Analysis



PCA:

Orthogonal projection of data onto 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)

Principle Components Analysis

Idea:

- Given data points in a d-dimensional space,
  project into lower dimensional space while preserving as much information as possible
  - Eg, find best planar approximation to 3D data
  - Eg, find best 12-D approximation to 10^4-D data

- In particular, choose projection that
  minimizes squared error
  in reconstructing original data

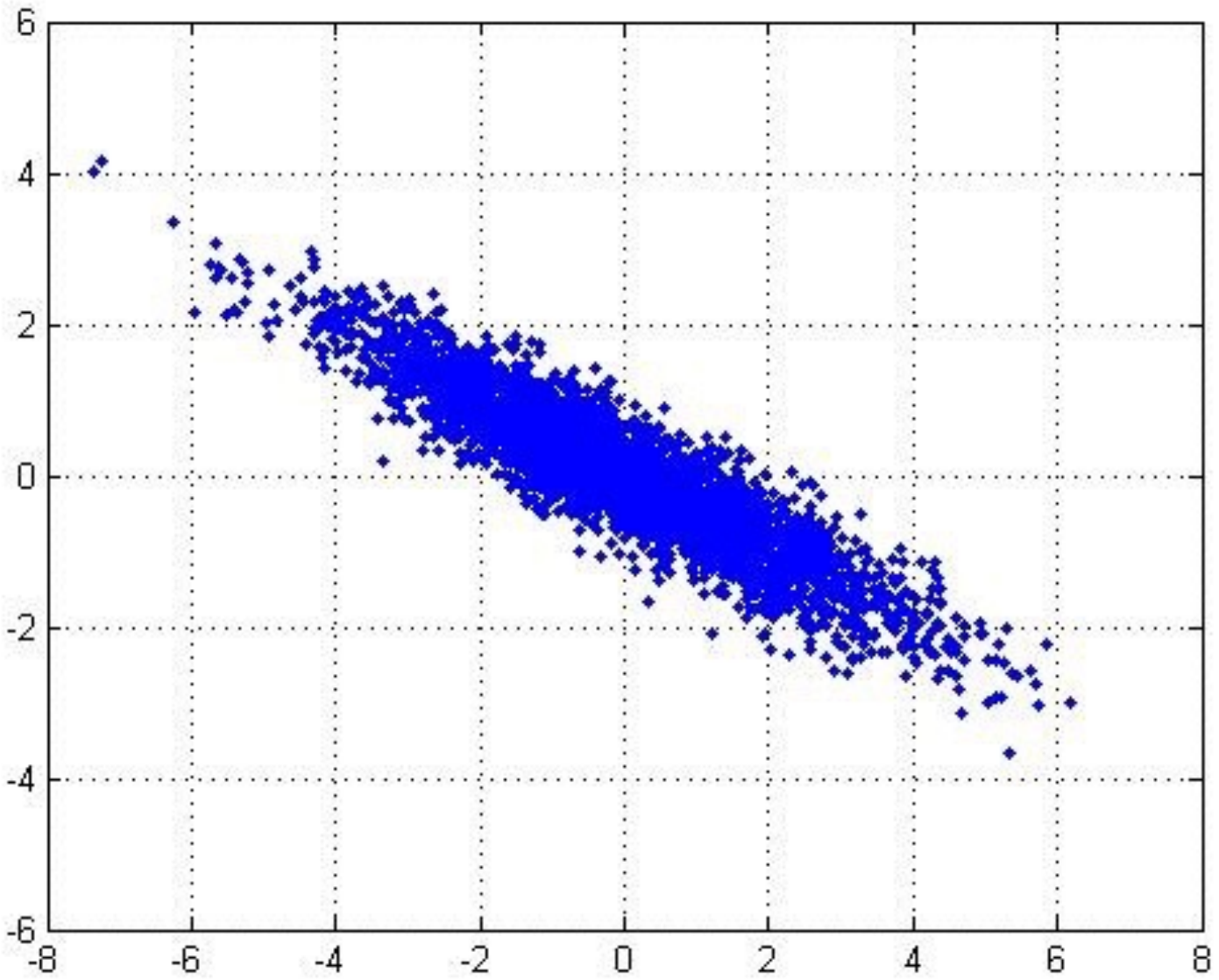
The Principal Components

- **Vectors** originating from the center of mass

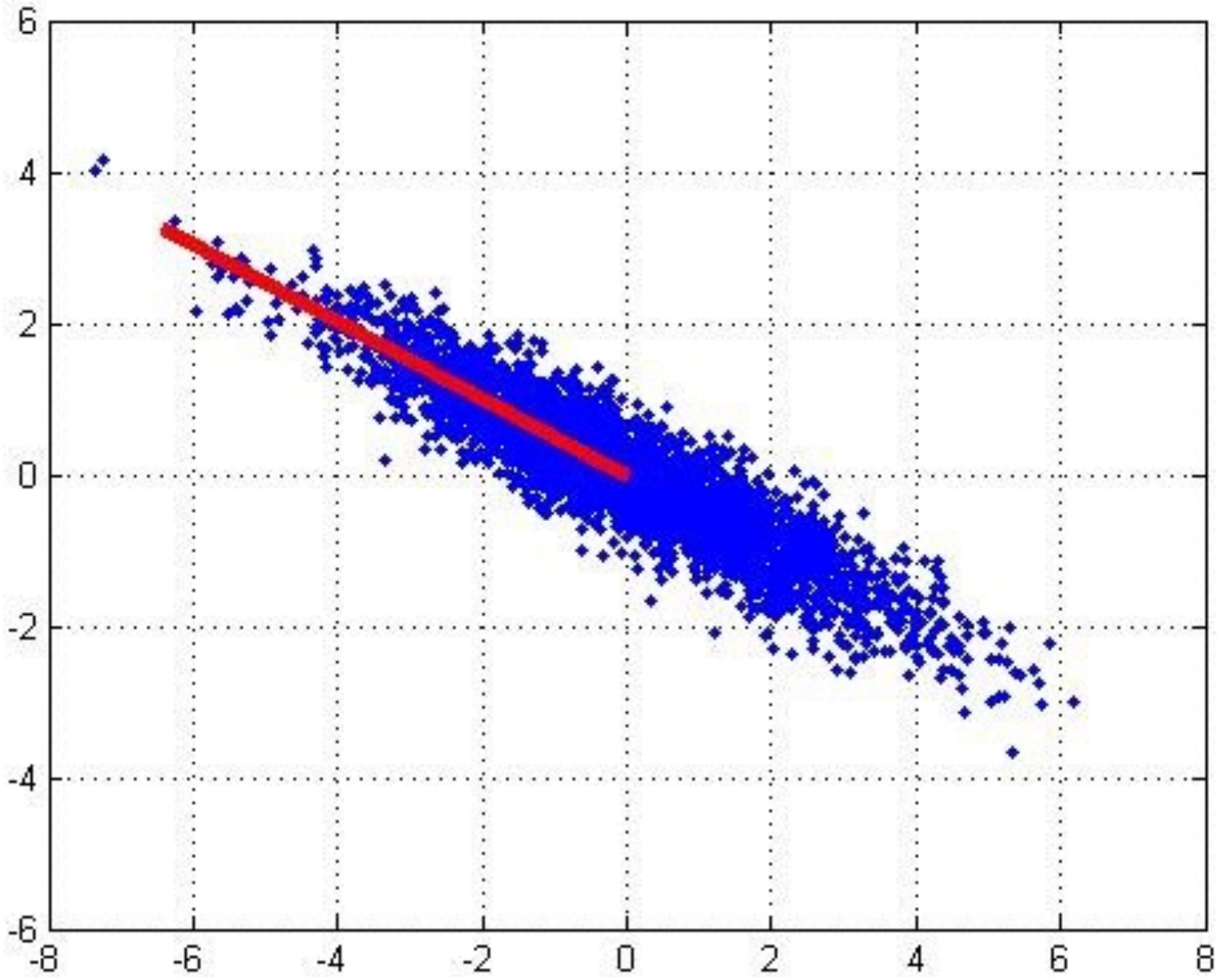
- Principal component #1 points in the direction of the **largest variance**.

- Each subsequent principal component…
  - is **orthogonal** to the previous ones, and
  - points in the directions of the **largest variance of the residual subspace**

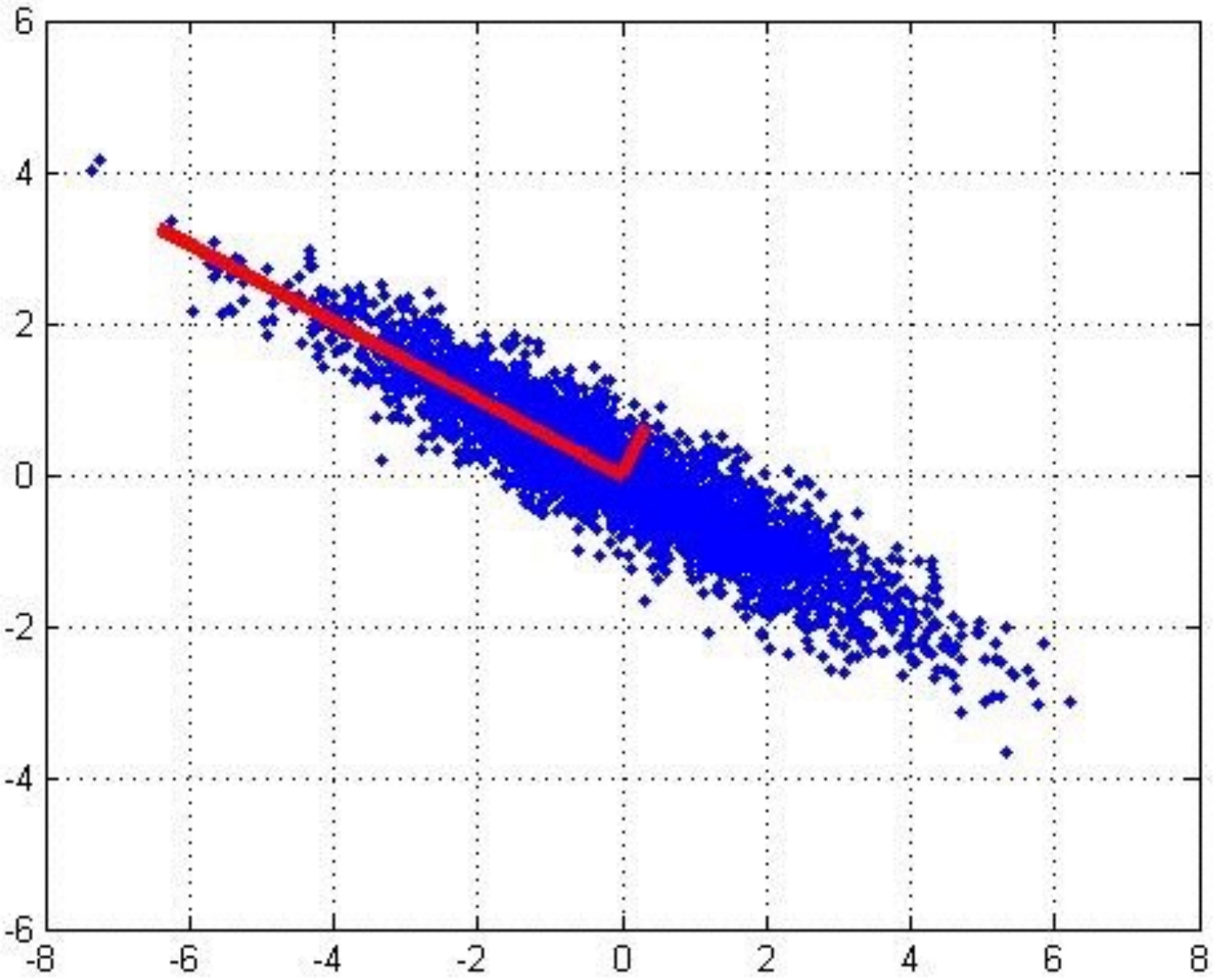
2D Gaussian dataset



1st PCA axis



2nd PCA axis



PCA algorithm I (sequential)

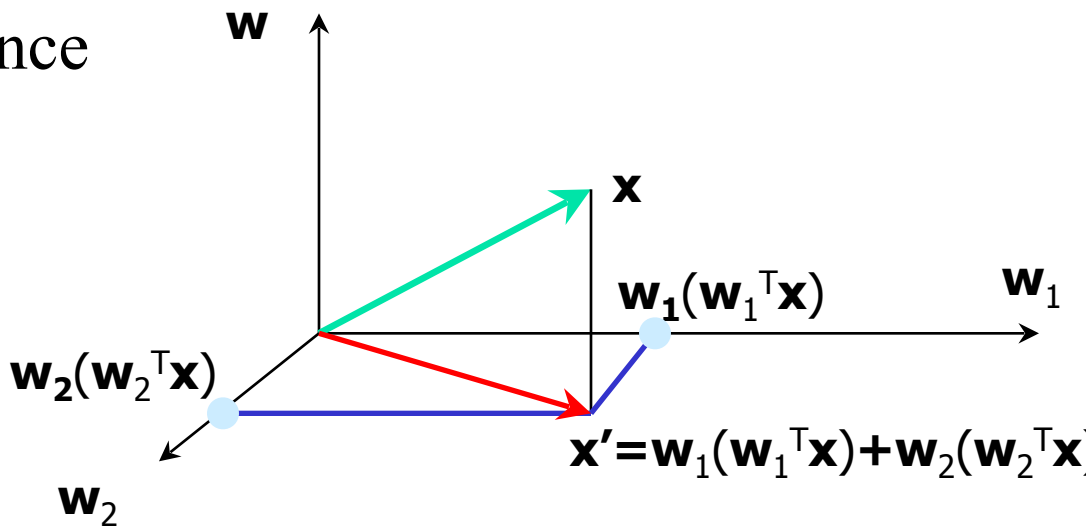
Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

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

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

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

We maximize the variance of the projection in the residual subspace



PCA algorithm II (sample covariance matrix)

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

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

- **PCA** basis vectors = the eigenvectors of Σ

- Larger eigenvalue $\Rightarrow$ more important eigenvectors

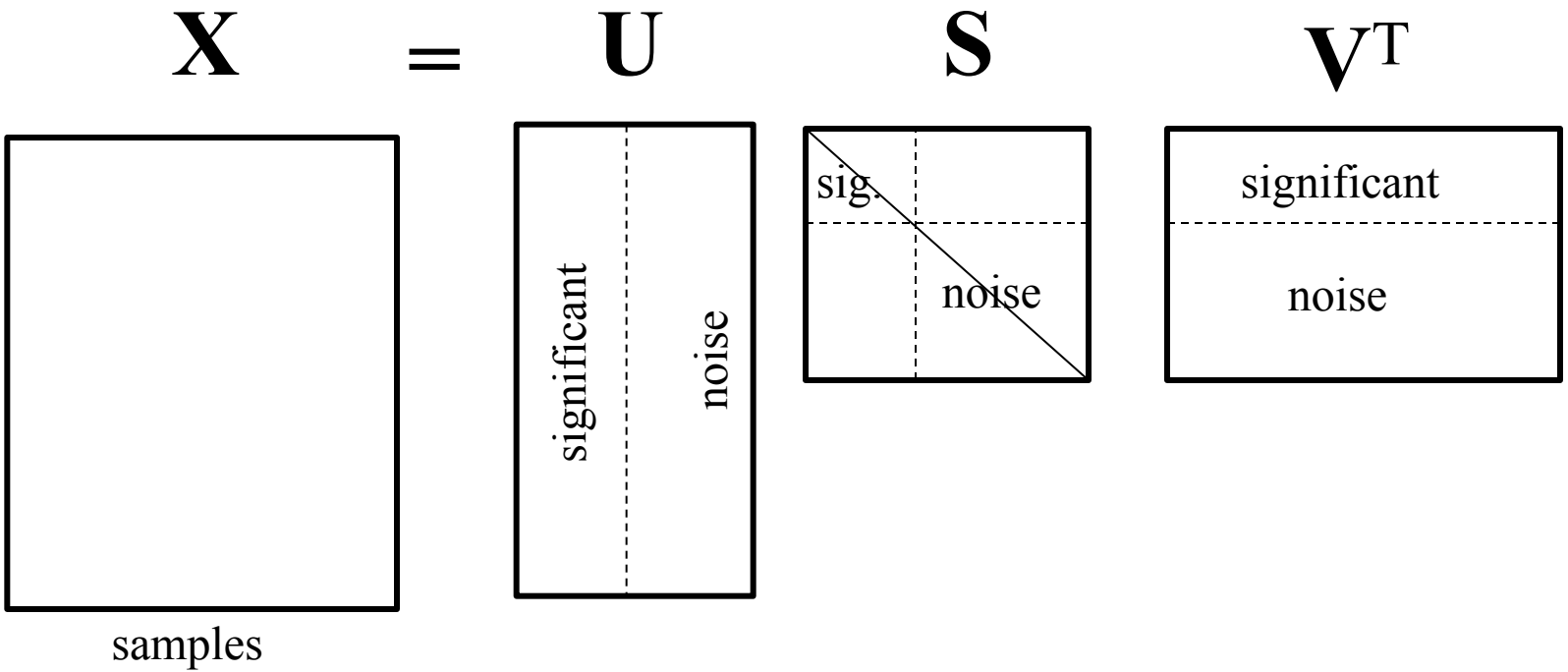
PCA algorithm III (SVD of the data matrix)

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

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



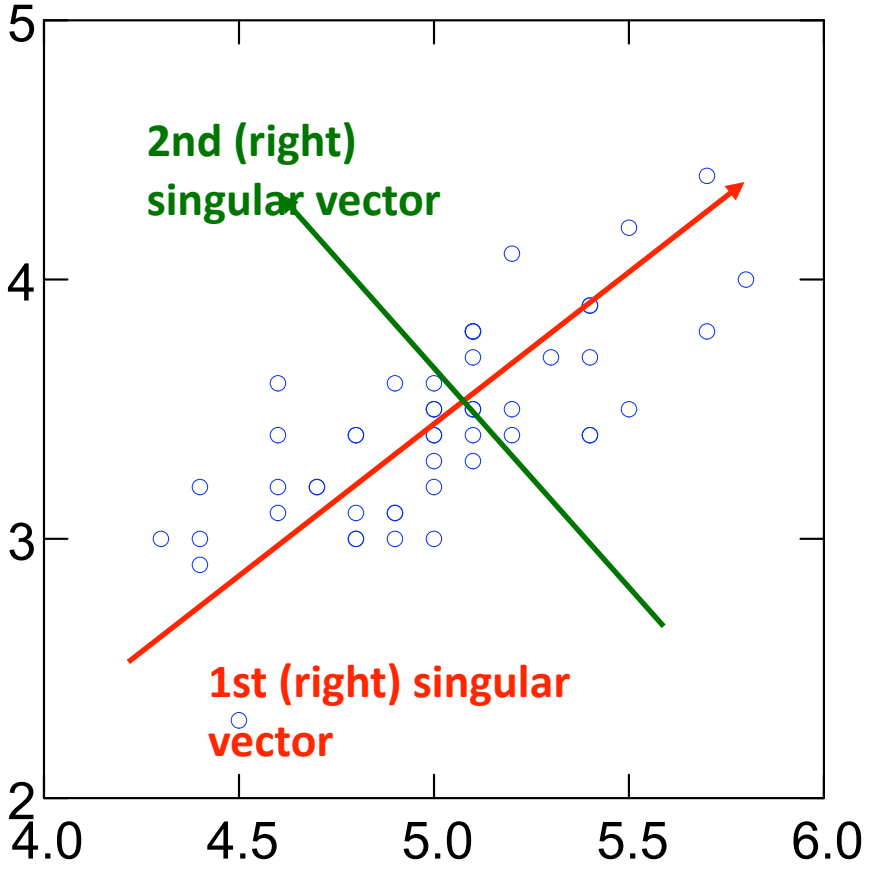
PCA algorithm III

- **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

More SVD: Example



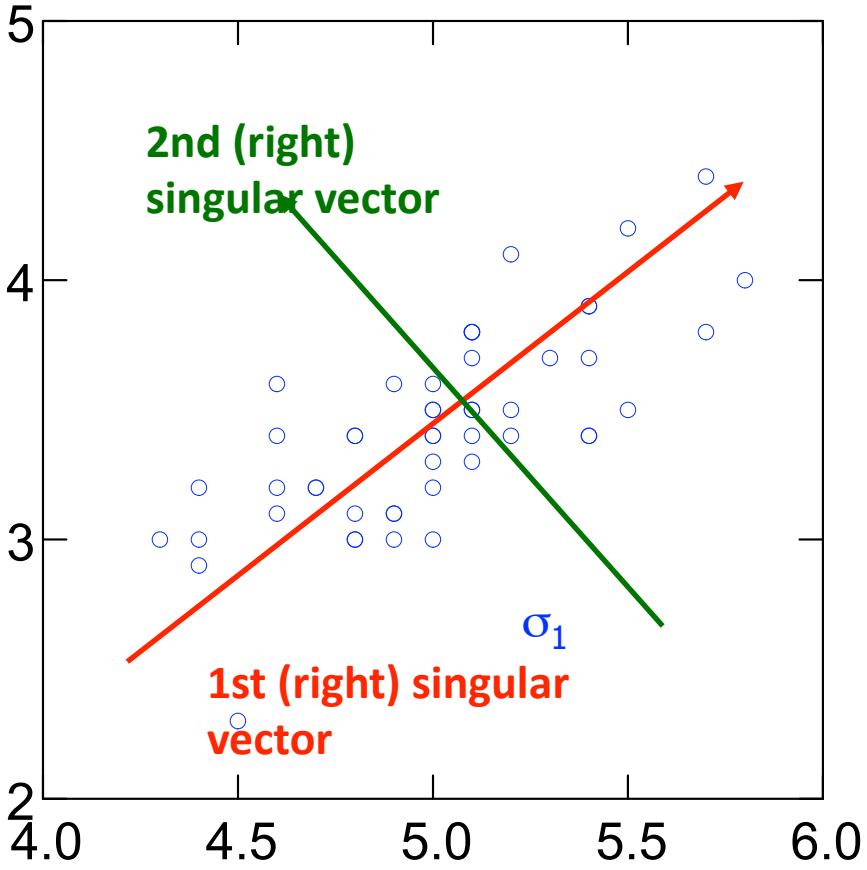
Input: 2-d dimensional points

Output:

1st (right) singular vector: direction of maximal variance,

2nd (right) singular vector: direction of maximal variance, after removing the projection of the data along the first singular vector.

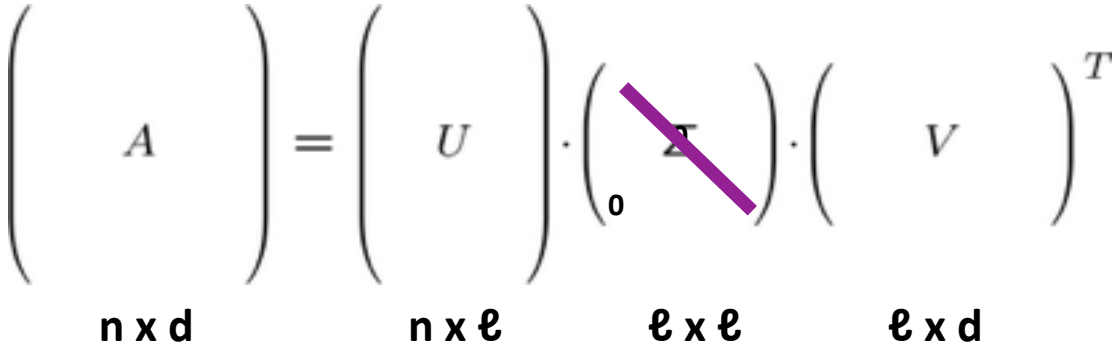
Singular values



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

SVD decomposition



U (V): orthogonal matrix containing the left (right) singular vectors of **A.**

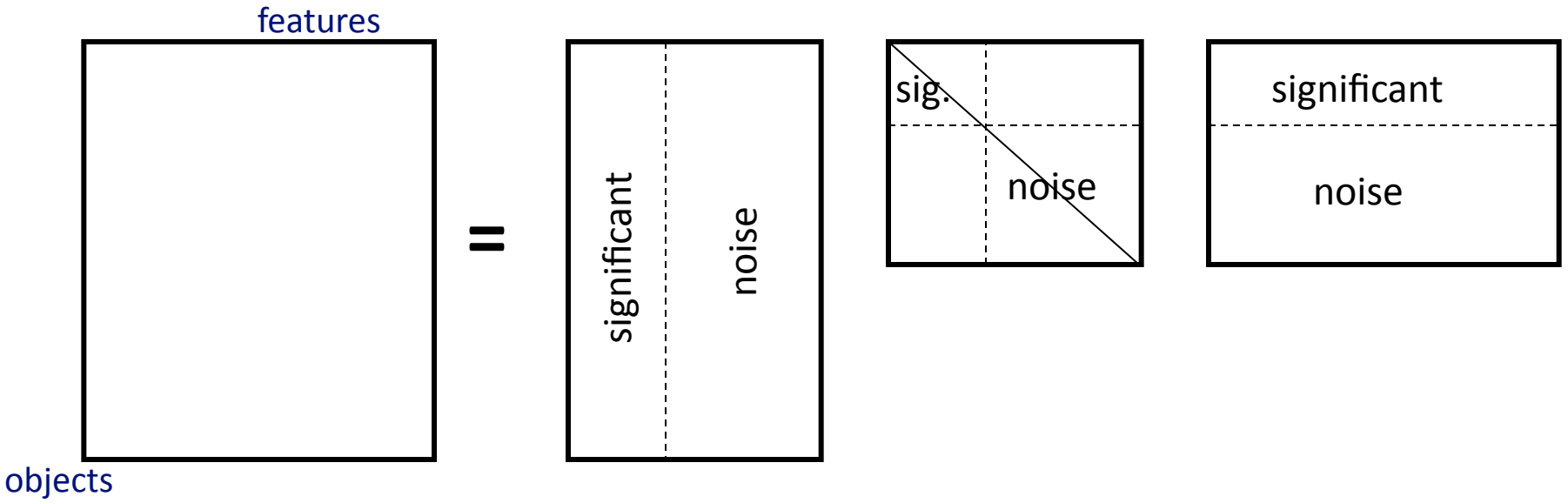
Σ: diagonal matrix containing the **singular values** of **A:**

($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\ell$ **)**

Exact computation of the SVD takes **O(min{mn^2 , m^2n})** time.
The top *k* left/right singular vectors/values can be ***computed faster*** using Lanczos/Arnoldi methods.

SVD and Rank-k approximations

$$A = U \Sigma V^T$$



Rank-k approximations (A_k)

$$\left(A_k \right) = \left(U_k \right) \cdot \left(\Sigma_k \right) \cdot \left(V_k^T \right)$$

n x d n x k k x d

U_k (V_k): orthog
singular vecto
Σ_k: diagonal ma

A_k is an approximation of A

A_k is the **best** approximation of A

PCA and SVD

- PCA is SVD done on **centered** data
- PCA looks for such a direction that the data projected to it has the maximal variance
- PCA/SVD continues by seeking the next direction that is orthogonal to all previously found directions
- All directions are orthogonal

How to compute the PCA

- Data matrix **A**, ***rows = data points***, ***columns = variables*** (attributes, features, parameters)

1. Center the data by subtracting the mean of each column
2. Compute the SVD of the centered matrix **A'** (i.e., find the first **k** singular values/vectors) $\mathbf{A' = U\Sigma V^T}$
3. The principal components are the columns of **V**, the coordinates of the data in the basis defined by the principal components are **UΣ**

Singular values tell us something about

- The variance in the direction of the **k**-th principal component is given by the corresponding singular value σ_k^2

- Singular values can be used to estimate how many components to keep

- ***Rule of thumb:*** keep enough to explain ***85%*** of the variation:

$$\frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{n} \sigma_j^2} \approx 0.85$$

SVD is “the **Rolls-Royce** and the **Swiss Army Knife** of Numerical Linear Algebra.”*

*Dianne O’Leary, MMDS ’06

Random Projections

- What happens if the dimensionality is very high and the number of points is large?
- SVD computation can be expensive (impossible?)

- We have another approach…

Closeness: Pairwise distances

- **Johnson-Lindenstrauss lemma:** Given **$\varepsilon>0$**, and an integer **n**, let **k** be a positive integer such that **$k \geq k_0 = O(\varepsilon^{-2} \log n)$**. For every set **X** of **n** points in **R^d** there exists **$F: R^d \rightarrow R^k$** such that for all **$x_i, x_j \in X$**

$$(1-\varepsilon)||x_i - x_j||^2 \leq ||F(x_i) - F(x_j)||^2 \leq (1+\varepsilon)||x_i - x_j||^2$$

What is the intuitive interpretation of this statement?

JL Lemma: Intuition

- Vectors $x_i \epsilon R^d$, are projected onto a **k**-dimensional space (**k<<d**): $y_i = R\, x_i$
- If $||x_i||=1$ for all **i**, then,

  $||x_i-x_j||^2$ is approximated by $(d/k)||x_i-x_j||^2$
- **Intuition:**
  - The expected squared norm of a projection of a unit vector onto a random subspace through the origin is **k/d**
  - The probability that it deviates from expectation is very small

JL Lemma: Why is true?

- $x=(x_1,\ldots,x_d)$, **d** independent Gaussian N(0,1) random variables; $y = 1/|x|(x_1,\ldots,x_d)$
- **z :** projection of **y** into first **k** coordinates
  - $L = |z|^2$, $\mu = E[L] = k/d$
- $Pr(L \geq (1+\varepsilon)\mu) \leq 1/n^2$ and $Pr(L \leq (1-\varepsilon)\mu) \leq 1/n^2$
- $f(y) = sqrt(d/k)z$
- What is the probability that for pair **(y,y')**: $|f(y)-f(y')|^2/(|y-y'|)$ ***does not*** lie in range $[(1-\varepsilon),(1+\varepsilon)]$?
- What is the probability that some pair suffers?

Finding random projections

- Vectors $\mathbf{x_i \epsilon R^d}$, are projected onto a **k**-dimensional space (**k<<d**)
- Random projections can be represented by linear transformation matrix **R**
- $\mathbf{y_i = R\, x_i}$

- What is the matrix **R**?

Finding matrix **R**

- Elements **R(i,j)** can be Gaussian distributed
- Achlioptas has shown that the Gaussian distribution can be replaced by R(i,j) $*\sqrt{3}$

$$R(i,j) = \begin{cases} +1 \text{ with prob } \frac{1}{6} \\ 0 \text{ with prob } \frac{2}{3} \\ -1 \text{ with prob } \frac{1}{6} \end{cases}$$

- All zero mean, unit variance distributions for **R(i,j)** would give a mapping that satisfies the ***JL*** lemma

- **Why is Achlioptas result useful?**
  - **Hint: sparse matrix, only +1, 0, -1 (easy operations)**