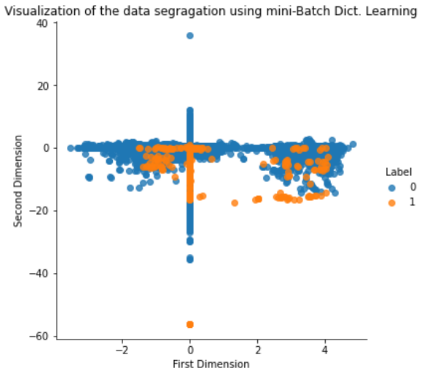
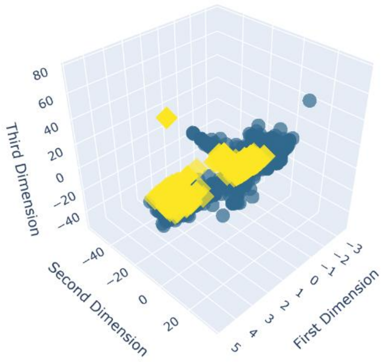
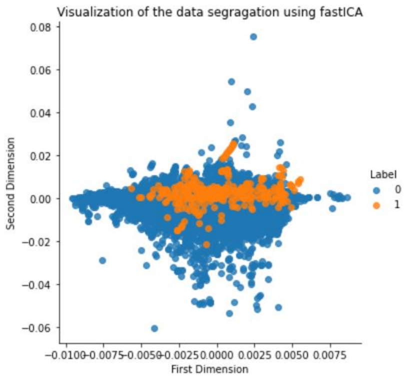
Data Driven Engineering I: Machine Learning for Dynamical Systems

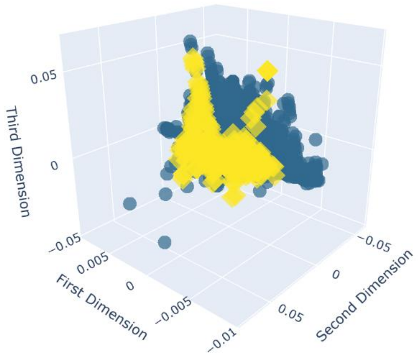
Analysis of Static Datasets II: Dimensionality Reduction

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer









Today's Agenda

Basic Steps to Follow:

0.) Understand the business/task.
1.) Understand the data.
2.) Explore & prepare the data.
3.) Shortlist candidate models.
4.) ~~Training the model~~
5.) Evaluate the model predictions
6.) "Serve" the model!

Still valid

2 major type

3 evaluation tools

Dimensionality Reduction

* When: Data has large number of features (dimensions)

① Computational: compress initial data as a preprocessing step

- e.g. k-Means $\propto (M \times N) \Rightarrow (M' \times N)$ $M' \ll M$

② Feature Extraction: lower dim. representation of the physics

- $M' < M \Rightarrow$ more effective usage of features
- $M' \approx M \Rightarrow$ Coordinate Transformation: $[x, y, z, u, v, w] \Rightarrow [PC_1, PC_2, u', v', w']$

Dimensionality Reduction

* When: Data has large number of features (dimensions)

  (3) Visualization : exploratory analysis of data (planning phase)

  - M → 2 // 3 space

Two major branches:

(i) Linear Projection methods
- eg. SVD, PCA, random projection

(ii) Non-linear projection (manifold learning)
- learn the curved distance
- isomap, MDS, LLE, t-SNE, ICA dictionary learning, Random trees embedding

#0 Understanding the task

- **Problem**: Manufacturing error in a production line
- **Modified sensory input:** 28 variables including sensory input
- 280,000 instances, where only a **small fraction** (~500) of products are **defective**.
- **Heuristic**: <0.5% is defective

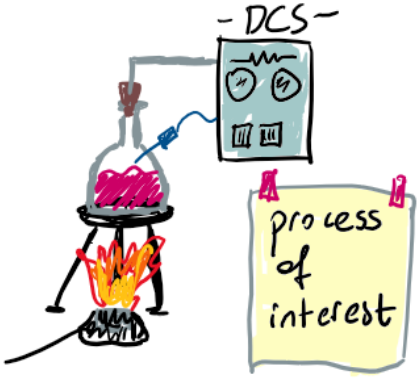
A similar example for you:

"Bosch Production Line Performance
Reduce manufacturing failures"

Dim. Reduction:

- Computational
  -preprocessing-
- Feature Extraction
  ~ pattern recognition ~
- Visualization

Idea:



$$\Rightarrow X = \begin{bmatrix} \text{— — —} \\ \text{— product}_n \text{ —} \\ \text{— — —} \end{bmatrix}$$

→ information m on the production line

Idea:

Assume:

- $product = \sum_{i=1}^{K} process_i$
- Features m is correlated to K steps in the production line;

Then:

$$X \rightarrow \boxed{D.R.} \rightarrow \tilde{X} \rightarrow \boxed{DR^T} \rightarrow X'$$

physical coordinates — "latent space" — physical coordinates

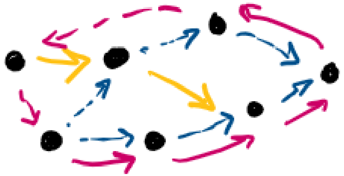
$\underbrace{\text{Encoding} \qquad \text{Decoding}}$

if DR is capable of learning the patterns in the physical system;

$$X \approx X'$$

Idea:

Interpreting Patterns



- Physical system is composed of logical steps;
- Logical steps followed ⇒ "Regular product"
- Failure at some point ⇒ "Defect"

"Outlier Detection"

A.I.

"Something is wrong here."

Aim ⇒ Learn enough to detect outliers;

#1 Understanding the data

- Check the data source: understand what the data refers to
- Objective: understand the characteristics of the data
- Look at the feature columns:
  - Any missing values?
  - Any features with NaN values?
  - Uniqueness of the dataset? (“cardinality”)

=> Colab

```
 23   S23      284807 non-null  float64
 24   S24      284807 non-null  float64
 25   S25      284807 non-null  float64
 26   S26      284807 non-null  float64
 27   S27      284807 non-null  float64
 28   S28      284807 non-null  float64
 29   Class    284807 non-null  object
dtypes: float64(29), object(1)
memory usage: 65.2+ MB
time: 54.5 ms
```

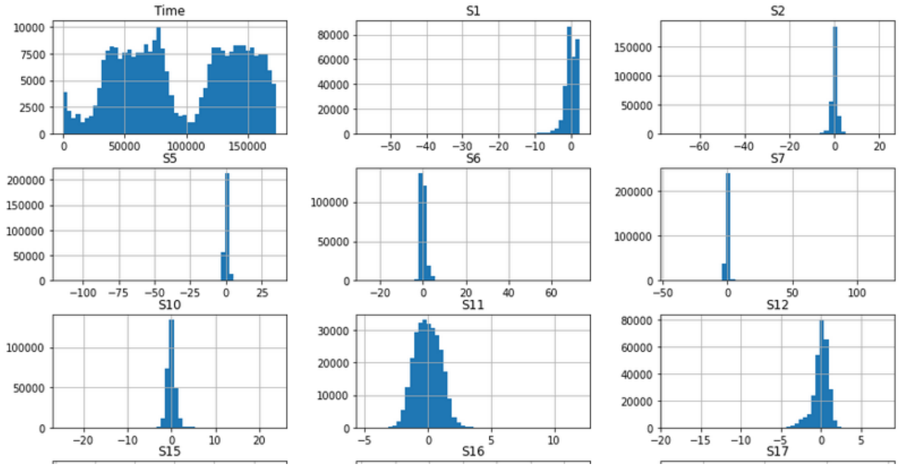
	Time	S1	S2	S3	S4	S5	S6
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	1.758743e-12	-8.252298e-13	-9.636929e-13	8.316157e-13	1.591952e-13	4.247354e-13
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01

```
time: 447 ms
```

#2 Exploring the data

- ❑ **Objective**: generate a data quality report

- ❑ Using standard statistical measures of central tendency and variation
  - ❑ tabular data and visual plots
  - ❑ mean, mode, and median
  - ❑ standard deviation and percentiles
  - ❑ bars, histograms, box and violin plots



- ✓ Missing values,
- ✓ Irregular cardinality problems,
  - ▪ 1 or comparably small
- ✓ Outliers
  - ▪ invalid outliers and valid outliers

#2 Exporing the data: Correlation Matrix

- Shows the correlation between each pair of features

$$Cov(a,b) = \frac{1}{n-1}\sum_{i=1}^{n}\left[(a_i - \bar{a}) \times (b_i - \bar{b})\right]$$

Features instance mean mean

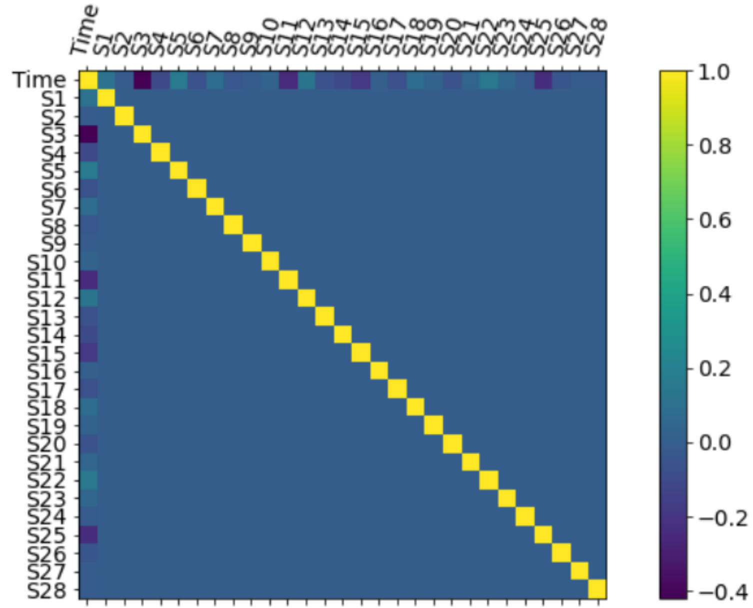
- Normalized form of “covariance”

$$\Downarrow$$

$$Corr(a,b) = \frac{Cov(a,b)}{SD(a) \times SD(b)}$$

* Normalized
* Dimensionless

Easy to interpret

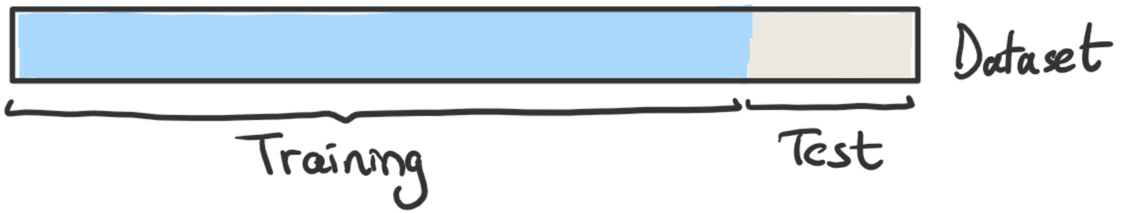


time: 975 ms

- Ranges between −1 and +1

#2 Preparing the Data

- Clustering >> unsupervised >> **training & test split not needed**



- We will use it to **reduce the volume of the data** when needed:

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(dataX,
     dataY, test_size=0.9,
     random_state=2020, stratify=dataY)

     time: 188 ms
```

#3 Candidate Models: PCA

Principal Component Analysis

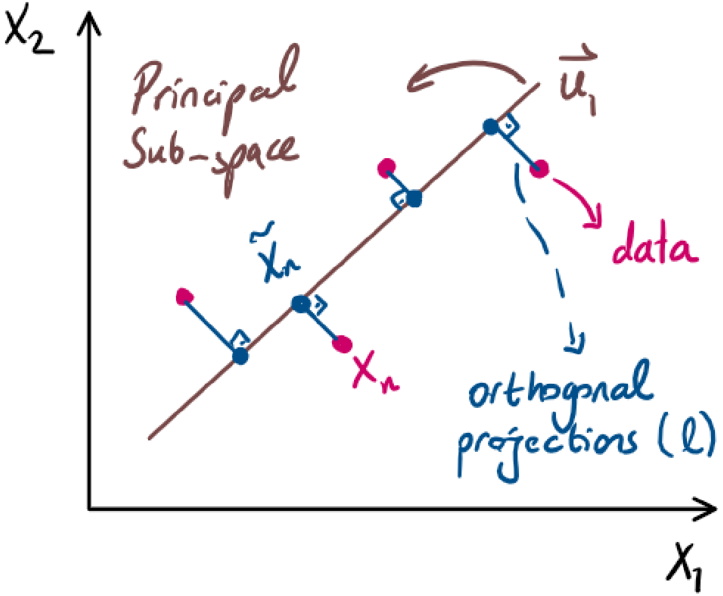
- [x] Looks into the correlation between features
- [x] Combines highly correlated ones.
- [x] New combined features ⇒ "Principal Components"

- Features $\rightarrow$ PC_i } Reconstruction is possible

Obj: minimum information loss

[info. ≡ Variance]

How PCA works?



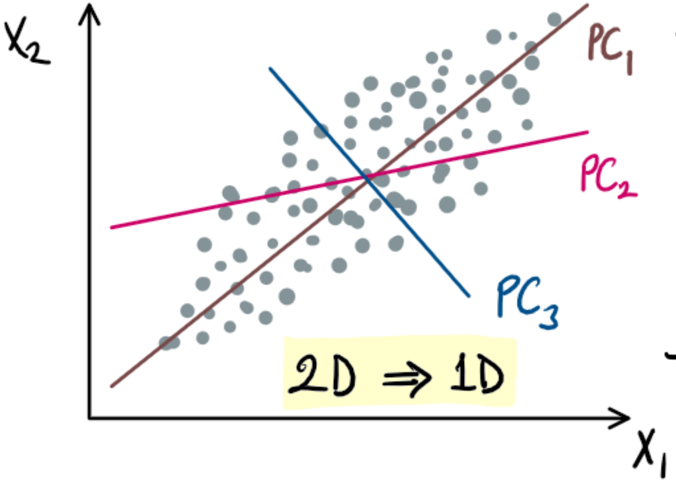
Objective:

* max. the variance of the • points — "maximum variance formulation"

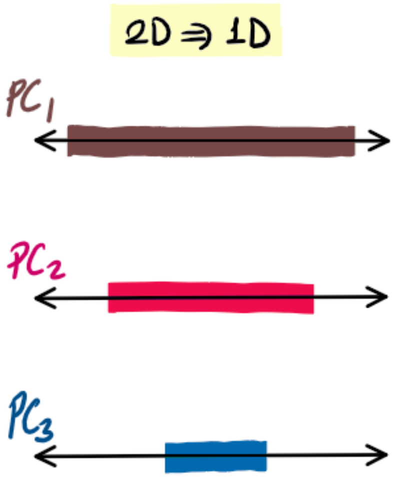
$\Updownarrow$

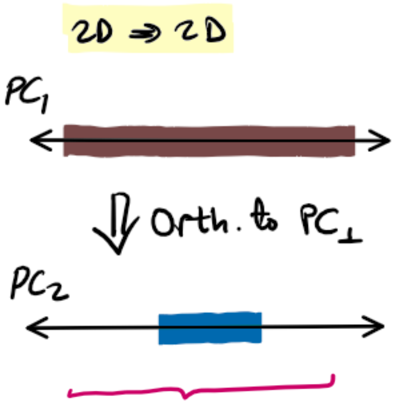
* Minimize the sum-of-squares of projection errors $\sum \ell_i$ — "minimum error formulation"

Max. Correlation: how does it work?



Checking Variance





For M Dimensions:

$[PC_1, PC_2, \ldots PC_M]$

Key Property of PCA: Hierarchical coordinate system

$PC_1 > PC_2 > PC_3 > \cdots > PC_M$

$$\Rightarrow \sum_{i=1}^{M'} PC_i \approx \sum_{i=1}^{M} PC_i$$

Solution Method: SVD

$$X = \begin{array}{c} \begin{array}{ccccc} & & & \text{feature } j & \end{array} \\ \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \end{array}$$

1. Evaluation of the mean $\bar{X}$
2. Finding covariance matrix S for dataset X.
3. Finding M' eigenvectors of S corresponding to M' largest eigenvalues.

Solution Method : SVD

① X must be scaled $\Rightarrow \bar{x}_j = 0$; $\underbrace{[-1, 1]}_{\text{whitened}}$

"mean centered data".

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

② Calculate the covariance matrix for data:

$$S = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T$$

③ Variance of the projected data on $\vec{u_1}$

$$\frac{1}{N} \sum_{n=1}^{N} \left\{ u_1^T x_n - u_1^T \bar{x} \right\}^2 = u_1^T S u_1$$

④ Maximize the projected variance wrt u_1 :

☒ Take derivative wrt u_1 ; equal to zero.

⇓

we need to prevent $\|u_1\| \to \infty$.

☑ Introduce a Lagrangian multiplier

⑤ $u_1^T S u_1 + \lambda_1 (1 - u_1^T u_1)$

⑥ $\frac{\partial}{\partial u_1} \to \emptyset \Rightarrow S u_1 = \lambda_1 u_1$

Solution Method: SVD

(7) $\|u_1^T \Rightarrow \boxed{u_1^T S u_1 = \lambda_1}$

$\Downarrow$

(8) Variance will be maximum when u_1 is equal to the eigenvector having the largest eigen value λ_1.

$\Downarrow$

"First principal component"

How PCA works?

* eigen-decomposition of the covariance matrix

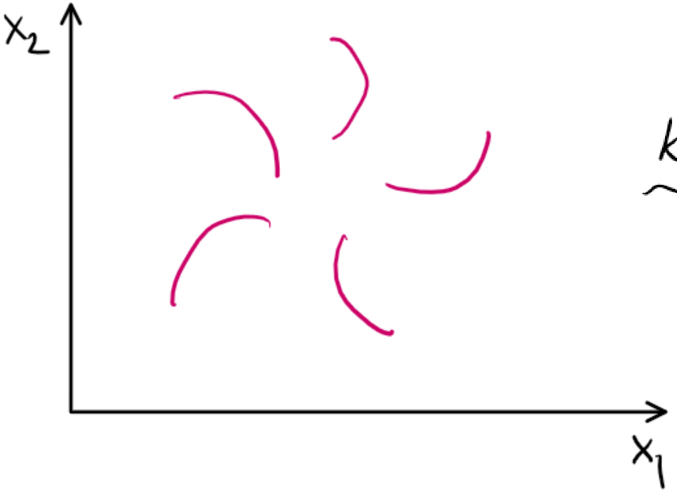
↳ PCs are orthogonal ⇒ uncorrelated to each other

↳ PCs have maximum correlation with measurements

#3 Candidate Models: kernel PCA

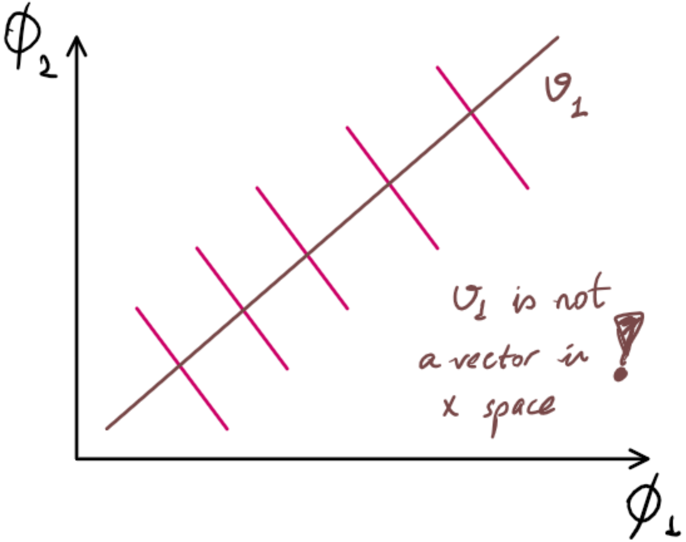
Karlsruhe Institute of Technology

kPCA



kernel trick:

$x \rightarrow \phi(x)$



#5 Evaluating the Results: Reconstruction error

$$X \rightarrow \underset{\text{Encoding}}{\boxed{D.R.}} \rightarrow \tilde{X} \rightarrow \underset{\text{Decoding}}{\boxed{DR^T}} \rightarrow X'$$

physical coordinates — "latent space" — physical coordinates

if DR is capable of learning the patterns in the physical system;

$$X \approx X'$$

* $loss = \sum_{m=1}^{M} (x_m - x'_m)^2 \Rightarrow N$ elements

Normalization:

* $loss' = \dfrac{loss - \min(loss)}{\max(loss) - \min(loss)} \Rightarrow [0, 1]$

Interpretation:

* $loss' \rightarrow 0 \Rightarrow$ Regular Product

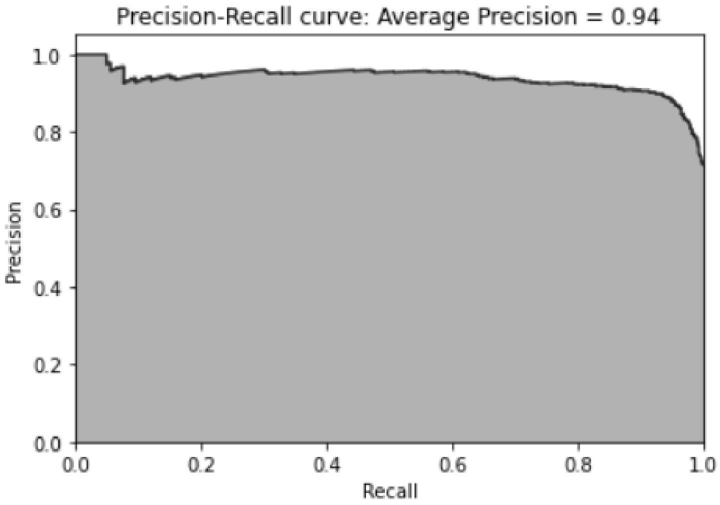
  $loss' \rightarrow 1 \Rightarrow$ Anomaly, defective

#5 Evaluation of the predictions

Precision Recall Curve (for imbalanced data)

$$\text{Precision} := \frac{\text{True Positive}}{\text{TP} + \text{False Positive}} \Rightarrow \frac{\text{It is positive}}{\text{“It is positive”}}$$

$$\text{Recall} := \frac{\text{True Positive}}{\text{TP} + \text{False Negative}} \Rightarrow \frac{\#\,\text{Correct Predict.}}{\#\,\text{True Cases}}$$



- **Precision** captures how often, when a model makes a positive prediction, this prediction turns out to be correct.
- **Recall** tells us how confident we can be that all the instances with the positive target level have been found by the model.

Ateliers & Saveurs in Montreal

colab

#3 Candidate Models: Dictionary Learning

Dictionary learning

* Obj: Sparse representation of original data

* Inspired from how visual cortex operates → image, sound, signal

- "Dict. Matrix" ← Sparse Matrices "atom"
- atom ← Binary vectors [0 0 1 .. 0 1]
- Each Instance := Weighted sum of atoms

} performs well for sparse systems

colab

#3 Candidate Models: ICA

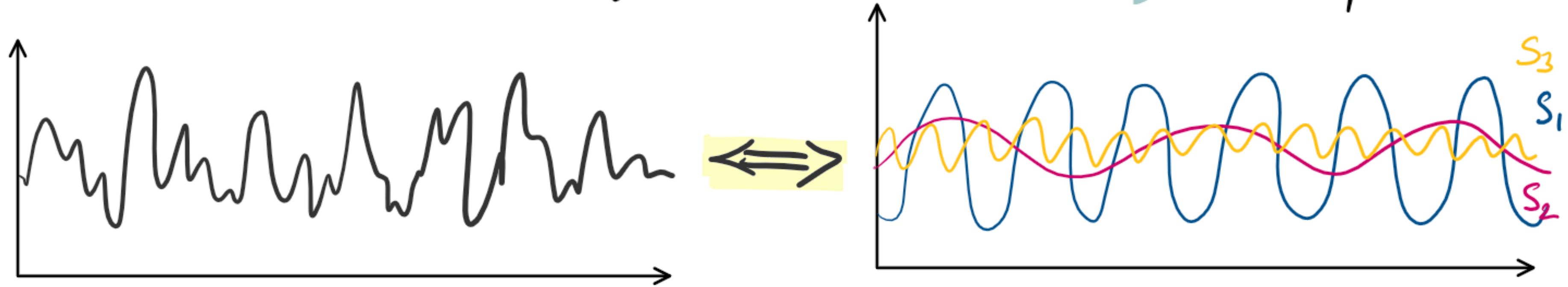
Karlsruhe Institute of Technology

Independent Component Analysis

* Bell & Sejnowski (1995)
* latent distribution is non-gaussian

Blind Source Separation

* Optical imaging
* Face recognition
* time series predictions
* gene expressions
* industrial processes



colab

#3 Candidate Models: Nonlinear Projections

① Multidimensional Scaling (MDS)

* Obj: preserve the pairwise distance between datapoints as closely as possible.

* Pairwise ⇒ Computationally expensive

* eigenvectors of "distance matrix"

* distance := Euclidean ⇒ "Expensive PCA"

#3 Candidate Models: Nonlinear Projections

② Locally Linear Embedding (LLE)

* Obj: preserve the distance with local neighbours

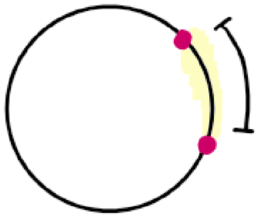
* Computes set of coeff. that best reconstruct the data from neighbouring points.

* Dimensions are reduced while preserving these coeff.

#3 Candidate Models: Nonlinear Projections

③ Isometric Feature Mapping (isomap)

* project data using MDS.
* uses geodesic distances;



arc length → distance

(i) First defines the neighbours for each data point.

(ii) List all neighb. points & distances (Euc.)

(iii) Find geodesic distances ($\sum_i$ arc_length$_i$)

(iv) MDS is applied.

#3 Candidate Models: Nonlinear Projections

④ Stochastic Neighbour Embedding (t-SNE)

* Obj. Convert the affinites of datapoints into joint probabilities.

* Good for identifying local structures.

Others ⇒ suitable for continuous manifolds.

* Good for visualizing high dimensional data.

(–) typically ~ $10^3 - 10^4$ times slower than PCA.

(–) Stochastic ⇒ Different seeds will give different clusters.

(–) Global structure may not be preserved if initiated randomly.

↳ you can intialize with PCA.

colab

Additional Notes

Content

(*) SVD & PCA

(i) PCA (ii) iPCA (iii) kPCA

(*) Anomaly Score

(*) PR-Curve

(*) 2D & 3D scatter plots.

(*) isomap

(*) LLE approach

(*) t-SNE

(*) MDS

reduce the dimensions.

(*) Dictionary Learning

(*) ICP