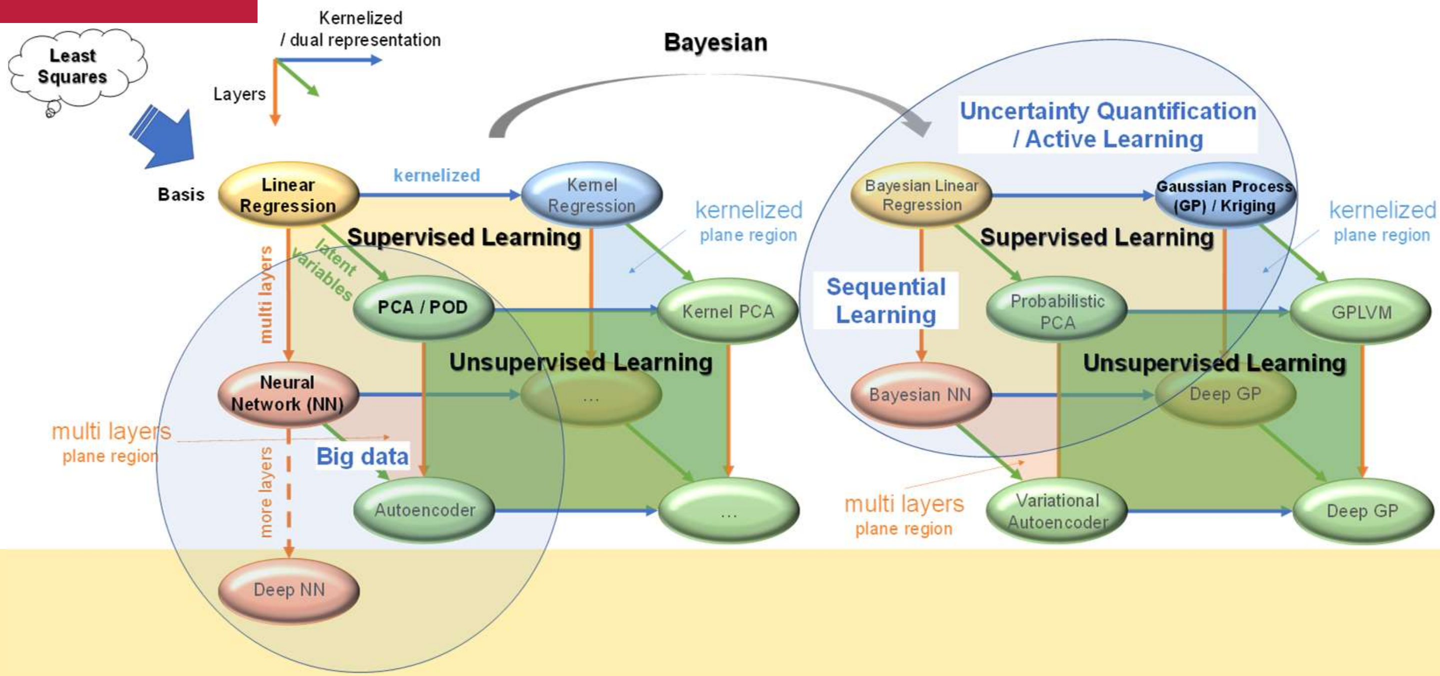


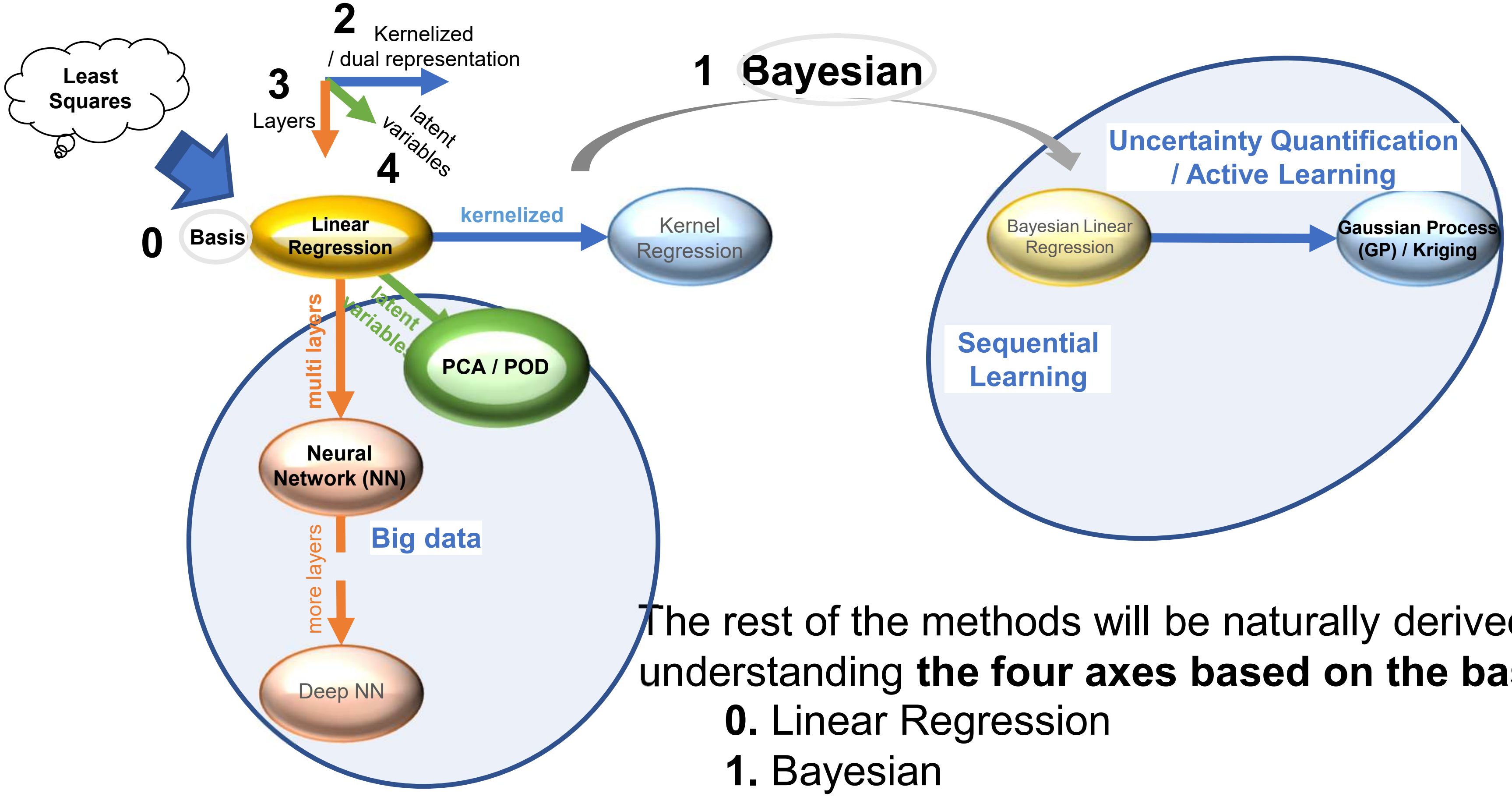
Scientific Machine Learning

Lecture 11: Unsupervised Learning

Dr. Daigo Maruyama

Prof. Dr. Ali Elham

Key Components (Repost from Lecture 1)



The rest of the methods will be naturally derived after understanding **the four axes based on the basis.**

0. Linear Regression
1. Bayesian
2. Kernel / Dual Representation
3. Layers
4. Latent Variables (Unsupervised Learning)

Finishing all the four axes here

Lecture content

- Unsupervised Learning (Concepts / Applications)
- Principal Component Analysis (PCA) / Proper Orthogonal Decomposition (POD) – Linear dimensionality reduction
- Nonlinear dimensionality reduction / Other methods

The lecture of this time partially follows the Chapters 12 of the book:
Christopher M. Bishop "Pattern Recognition And Machine Learning" Springer-Verlag (2006)
The name of this book is shown as “PRML” when it is referred in the slides.

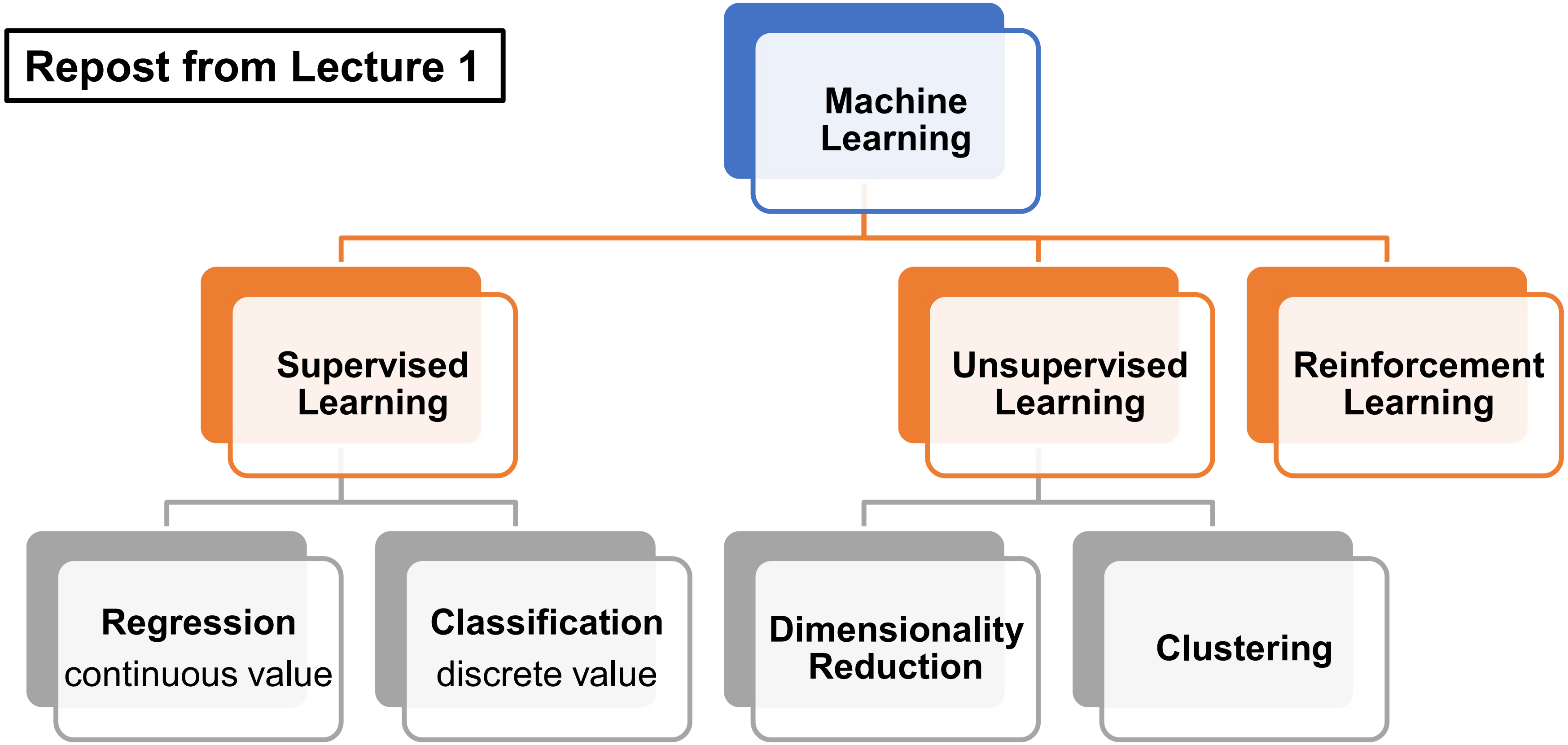
The lecture slides contains original topics in addition to the above.

Lecture content

- Unsupervised Learning (Concepts / Applications)

Machine Learning Classification by Use/Application

Repost from Lecture 1



In this course, machine learning classification is done by **methods and their concepts**.

Then the use/application is naturally derived/understood.

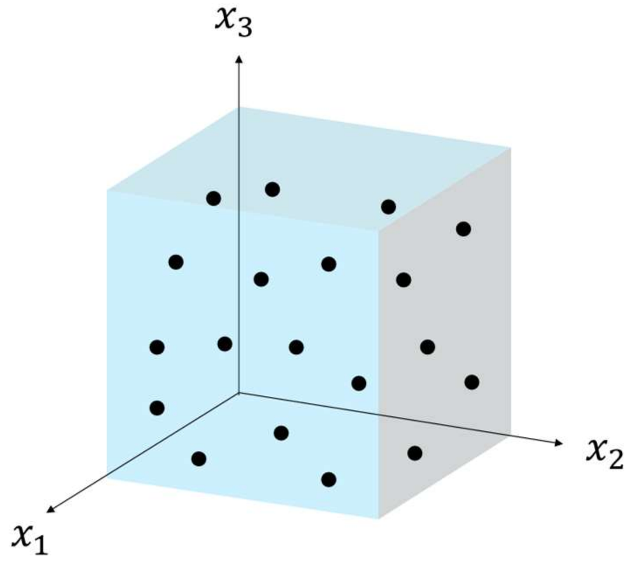
Concept / Motivation of Unsupervised Learning

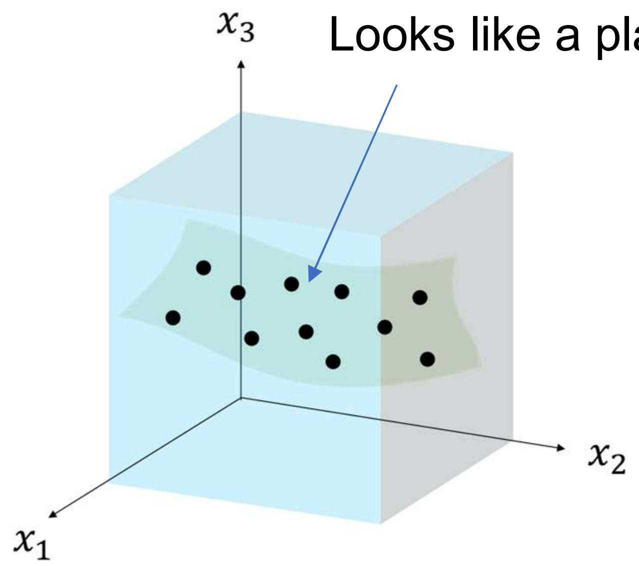
3D input space

collected pictures that look numbers

e.g. a sample generated by DoE

e.g. a sample selected <u>for some reasons</u>



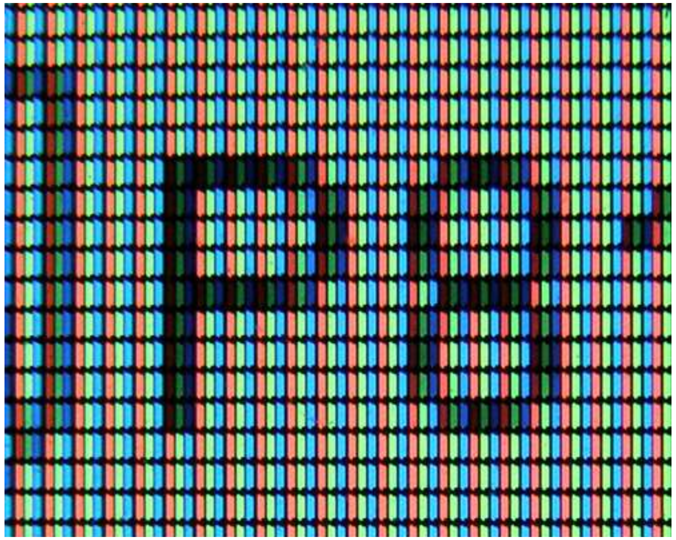


Looks nothing to do

Looks possible to be projected on a 2D space

Concept / Motivation of Unsupervised Learning

MNIST database



https://en.wikipedia.org/wiki/Pixel#/media/File:Closeup_of_pixels.JPG

Image data
= color (or brightness of black) data at each pixel
= **value at each coordinate**

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_{100\times 100})$$

10,000 dimensional input!

Concept / Motivation of Unsupervised Learning

Example of data

only two input parameters actually:

- Rotation
- Translation

$$\boldsymbol{x} = (x_1, x_2)$$

Inherently 2 dimensional input

Latent variables

Image data
= color (or brightness of black) data at each pixel
= **value at each coordinate**

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_{100\times100})$$

10,000 dimensional input!

Use / Applications

- **Dimensionality reduction**
  - various benefits on computational expense
    - GP models (lower sample size N as a result)
    - NN models

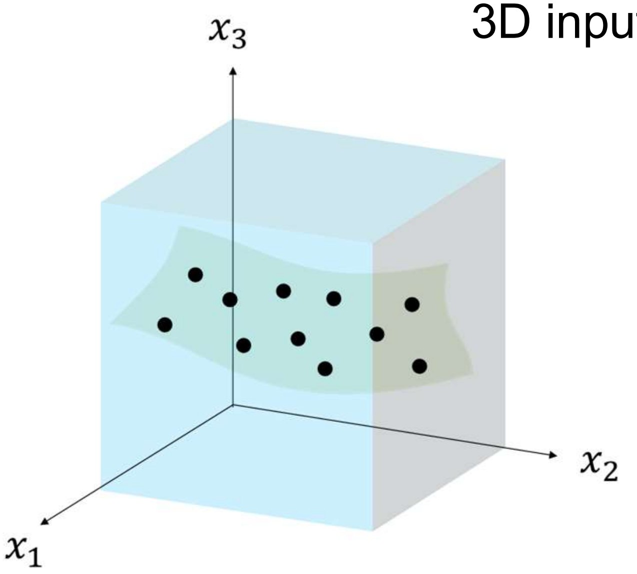
- **Feature extraction**
  - can possibly clarify the essence of the data
    - e.g. rotation and translation in the previous example
    - e.g. the contours of the objects in pictures

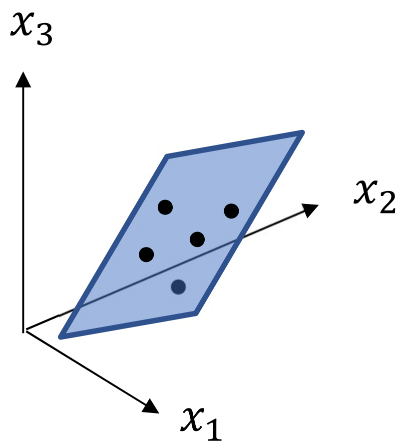
- **Visualization (2D or 3D)**

Examples of benchmark testcases
- MNIST
- Iris dataset

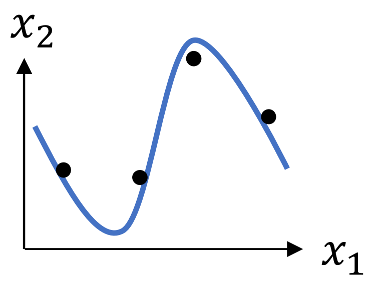
How can we extract the Latent Variables?

3D input space





2D input space



Have you ever seen something similar to this?

➡ the regression models (in Lecture 4)

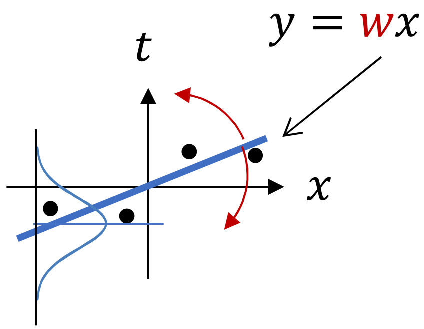
Supervised Learning in general

Lecture content

- Principal Component Analysis (PCA) / Proper Orthogonal Decomposition (POD) – Linear dimensionality reduction

Extension from Supervised Learning

Input-output space



the error defined on the output space t

Least Squares

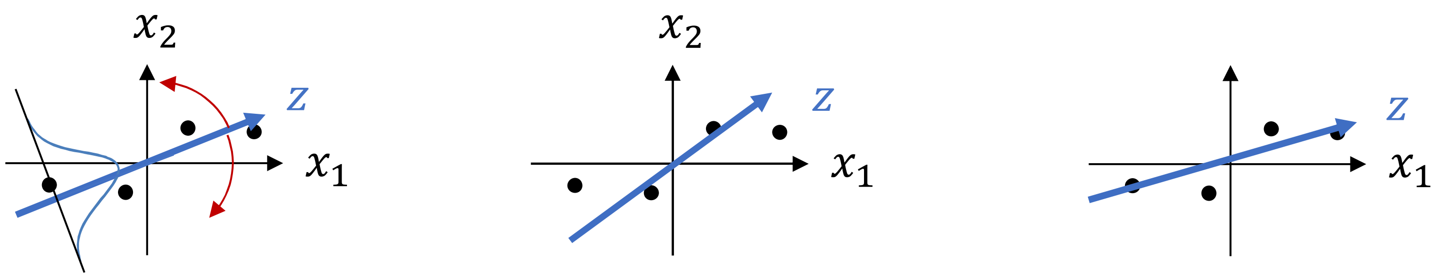
$$\min \qquad E = \sum_{i=1}^{N} \{t_i - \boldsymbol{w} x_i\}^2$$

w.r.t. $\boldsymbol{w}$

$\hat{\boldsymbol{w}}$

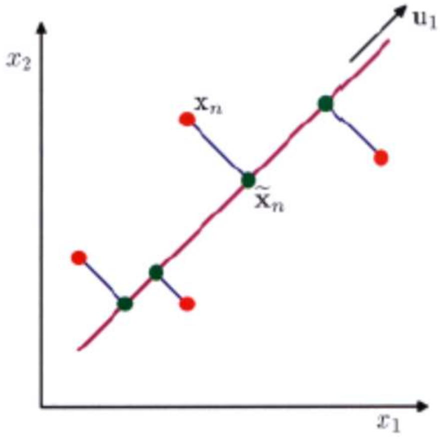
Input space

z is the new axis.



How we define the error (the probabilistic model)? Normal to z?

Extension from Supervised Learning



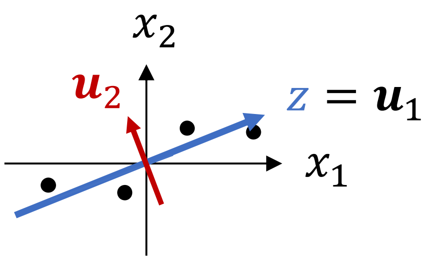
PRML, Fig. 12.2

consider the error on the axis normal to z

min $\quad E = \sum_{n=1}^{N} \{(x_{1n} - \tilde{x}_{1n})^2 + (x_{2n} - \tilde{x}_{2n})^2\}$

w.r.t. $\tilde{x}_1$, $\tilde{x}_2$

s.t. $\quad \tilde{x}_2 = w\tilde{x}_1$



z (as $\boldsymbol{u}_1$) has been determined.

$\min E$ is realized on the direction of:

the eigenvector $\boldsymbol{u}_2$ of the minimum eigenvalue of the covariance matrix $\mathbf{S}$ of the data $\mathbf{X}$

$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}$$

$$\mathbf{X} = \mathbf{X}_{org} - \boldsymbol{m}_{\mathrm{X}}$$

centered

Principal Component Analysis (PCA) / Proper Orthogonal Decomposition (POD)

Two commonly used definition:

- **Maximum variance formulation**
  - PRML, 12.1.1
- **Minimum-error formulation**
  - PRML, 12.1.2
  - Another explanation: introduced in the previous slides as an extension of the supervised learning techniques (simple linear regression models)

⇒ the eigenvalue problem

Linear algebra

of the covariance matrix $\boldsymbol{S}$ of the data $\mathbf{X}$

$$\boldsymbol{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}$$

Image of maximum variance formulation

$\mathbf{S} = \mathbf{U\Lambda U}^{-1}$

D: dimensionality of the input parameter $\boldsymbol{x}$
$D = 2$ in the previous example

$$\mathbf{\Lambda} = \begin{array}{c} \\ 1,\cdots,M \\ \left.\begin{pmatrix} \boxed{\begin{matrix}\lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots\end{matrix}} \\ \begin{matrix} 0 & \cdots & \lambda_D \end{matrix} \end{pmatrix}\right\} 1,\cdots,D \end{array}$$

λ_i: eigenvalue in **descending order**

$$\mathbf{U} = \begin{array}{c} 1,\cdots,M \\ \left.\begin{pmatrix} \boxed{\begin{matrix}\boldsymbol{u}_1 \\ \vdots\end{matrix}} \\ \boldsymbol{u}_D \end{pmatrix}\right\} 1,\cdots,D \end{array}$$

$\boldsymbol{u}_i$: eigenvector in the respective order of λ_i

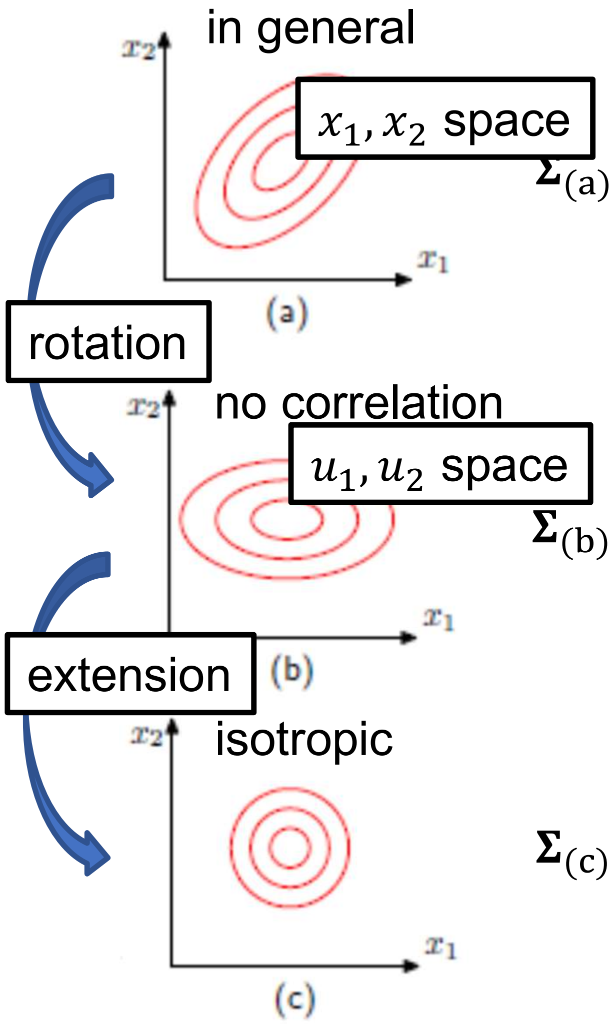
$M < D$ ➡ **Dimensionality reduction**

usually M is judged by the eigenvalues

All the equations can be naturally understood from the linear algebra.

$$\boldsymbol{x} = \sum_{i=1}^{M} (\boldsymbol{x}^{\mathrm{T}} \boldsymbol{u}_i) \boldsymbol{u}_i$$

Probability Distributions (Repost from Lecture 3)



$$\boldsymbol{\Sigma}_{(\mathrm{a})} = \begin{pmatrix} {\sigma_1}^2 & \cdots & \sigma_{D1} \\ \vdots & \ddots & \vdots \\ \sigma_{1D} & \cdots & {\sigma_D}^2 \end{pmatrix}$$

$\mathcal{N}(\mathbf{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$$\boldsymbol{\Sigma}_{(\mathrm{b})} = \begin{pmatrix} {\sigma_1}^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {\sigma_D}^2 \end{pmatrix}$$

$\mathcal{N}(y|\mu, {\sigma_i}^2)$ for $i = 1, \cdots, M$

for each axis

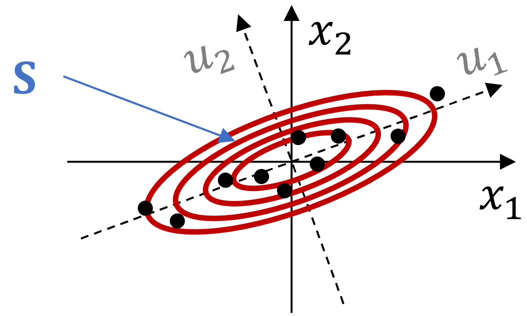
$$\boldsymbol{\Sigma}_{(\mathrm{c})} = \sigma^2 \mathbf{I} = \begin{pmatrix} \sigma^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma^2 \end{pmatrix}$$

$\mathcal{N}(y|\mu, \sigma^2)$

for each axis

Image of maximum variance formulation

connection between
the linear algebra and
the probability theory

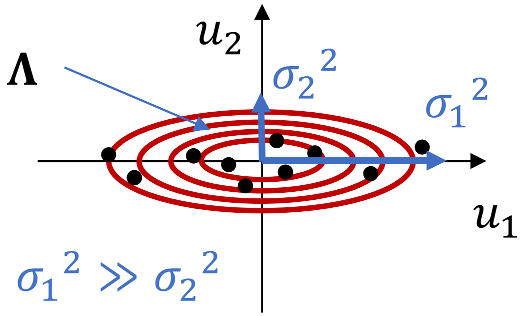


How the data $\mathbf{X}$
was generated.

$$
\begin{tikzcd}
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \arrow[rr, "\mathbf{S}"'] \arrow[d, "\mathbf{U}^{-1}"'] & & \mathbf{S}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\
\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \arrow[rr, "\mathbf{\Lambda}", "\text{scaled by } \sigma_i{}^2 \text{ in this space}"'] & & \mathbf{\Lambda}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \arrow[u, "\mathbf{U}"']
\end{tikzcd}
$$

$$\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$$

The data info on u_2
can be neglected.



$\sigma_1{}^2 \gg \sigma_2{}^2$

$$\mathbf{\Lambda} = \boxed{\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} = \begin{pmatrix} \sigma_1{}^2 & 0 \\ 0 & \sigma_2{}^2 \end{pmatrix}}$$

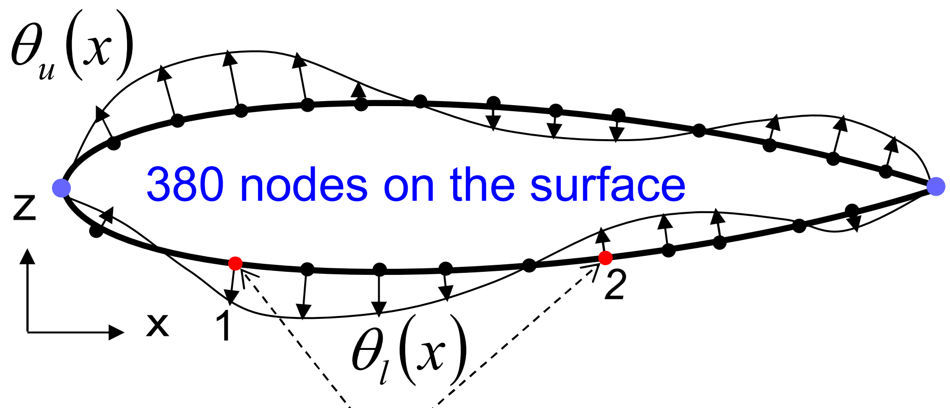
e.g. $\frac{\lambda_1}{\lambda_1+\lambda_2} \geq 90\%$

Example

$\boldsymbol{x} = (x_1, x_2, \cdots, x_{380})$

describing the airfoil shape by using <u>380</u> original parameters

objective: reduced



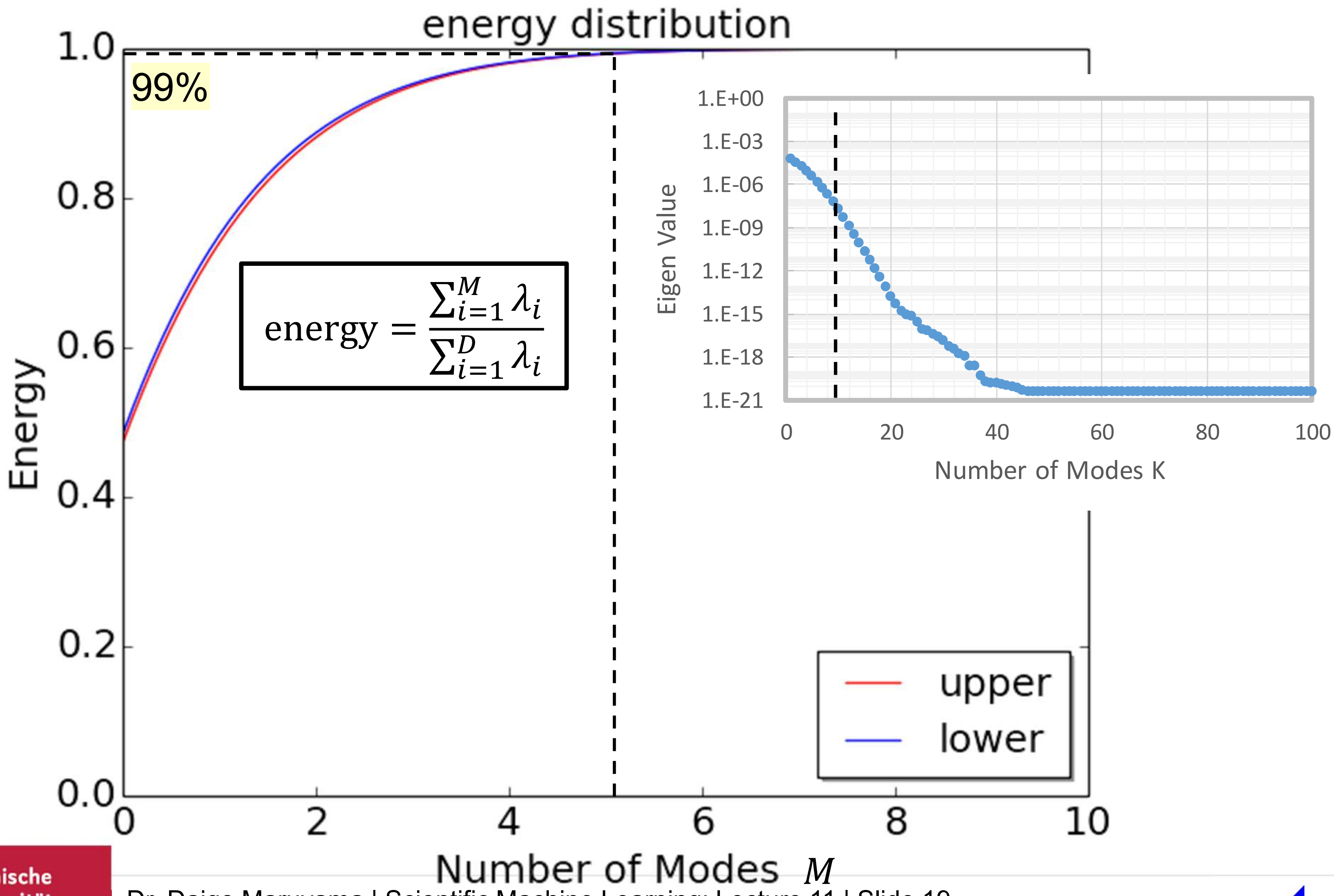
$$\mathrm{cov}[\theta_l(x_1), \theta_l(x_2)] = \sigma_\theta(x_1)\sigma_\theta(x_2)\exp\left(-\frac{(x_1 - x_2)^2}{l^2}\right)$$

The correlation was intentionally assumed to achieve plausible (smooth like the left sketch) surfaces.

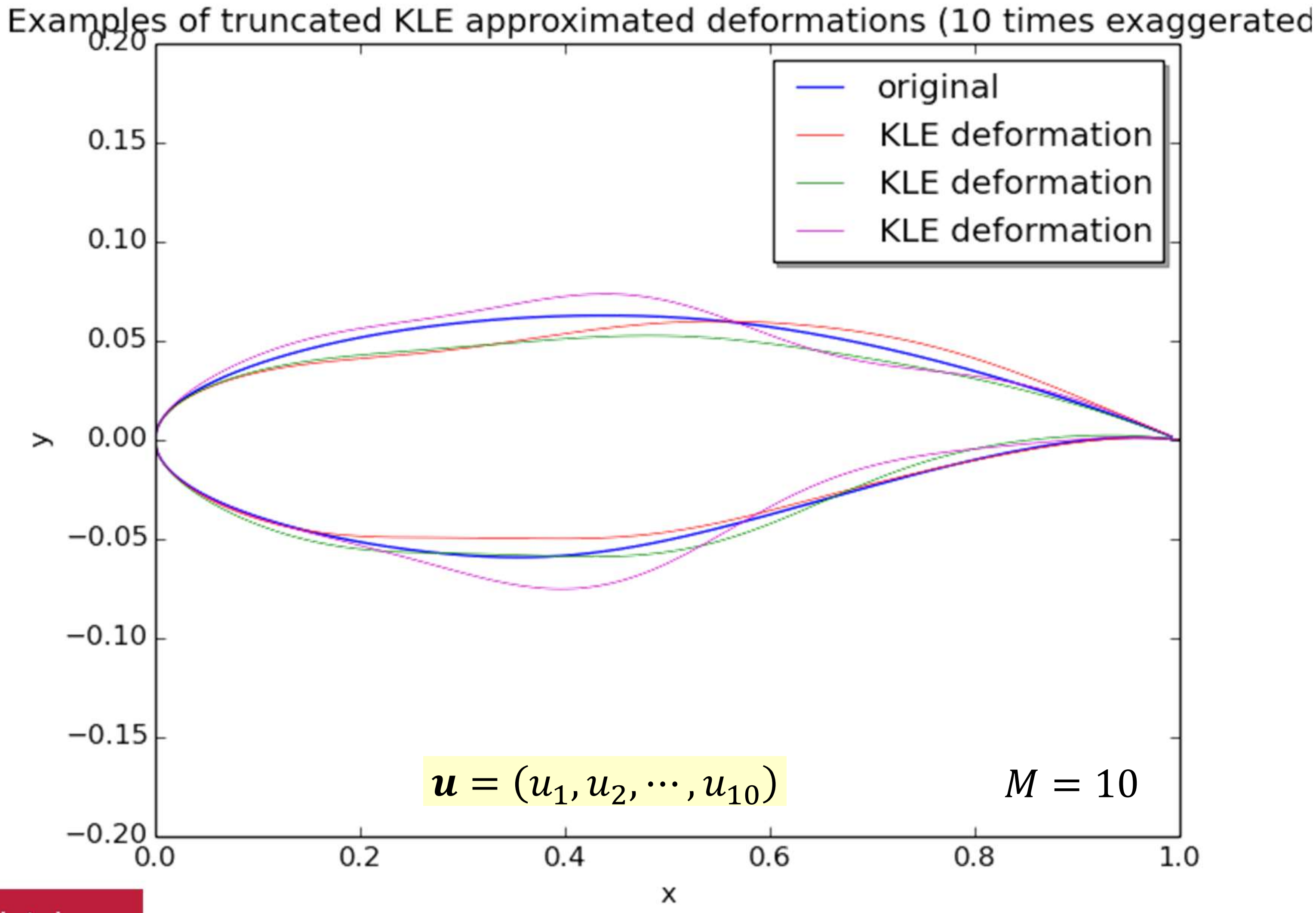
= can be also regarded as a GP model

Maruyama, D., Liu D., and Görtz S., "An efficient aerodynamic shape optimization framework for robust design of airfoils using surrogate models," in: Proceedings of the VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2016), 2016.

Example


energy distribution
99%
$$\text{energy} = \frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{D} \lambda_i}$$
Energy
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
Number of Modes M
upper
lower
Eigen Value
1.E+00
1.E-03
1.E-06
1.E-09
1.E-12
1.E-15
1.E-18
1.E-21
0
20
40
60
80
100
Number of Modes K

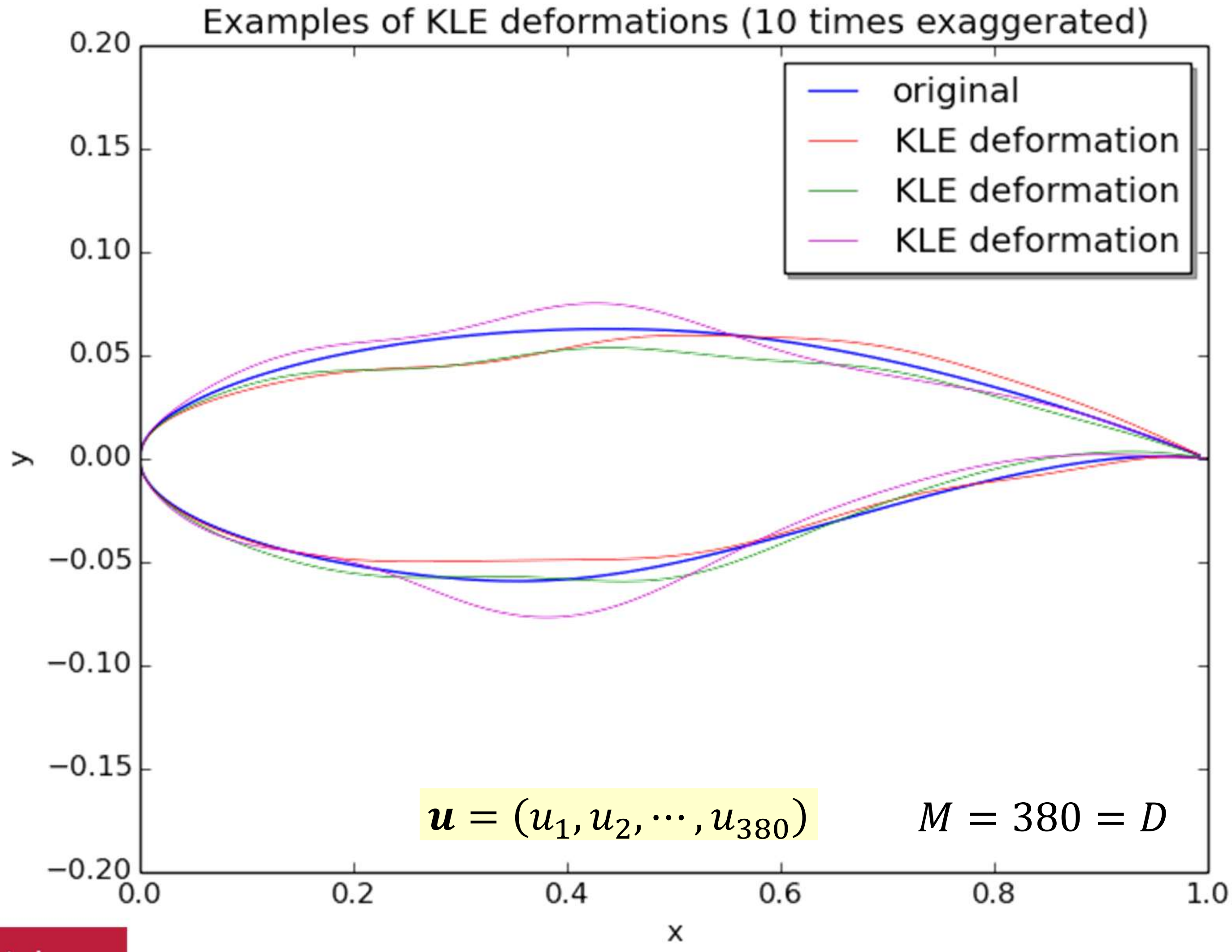
Example



Technische Universität Braunschweig

IFL

Example



$$\boldsymbol{u} = (u_1, u_2, \cdots, u_{380}) \qquad M = 380 = D$$

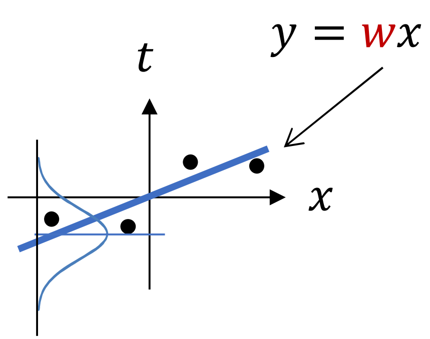
Lecture content

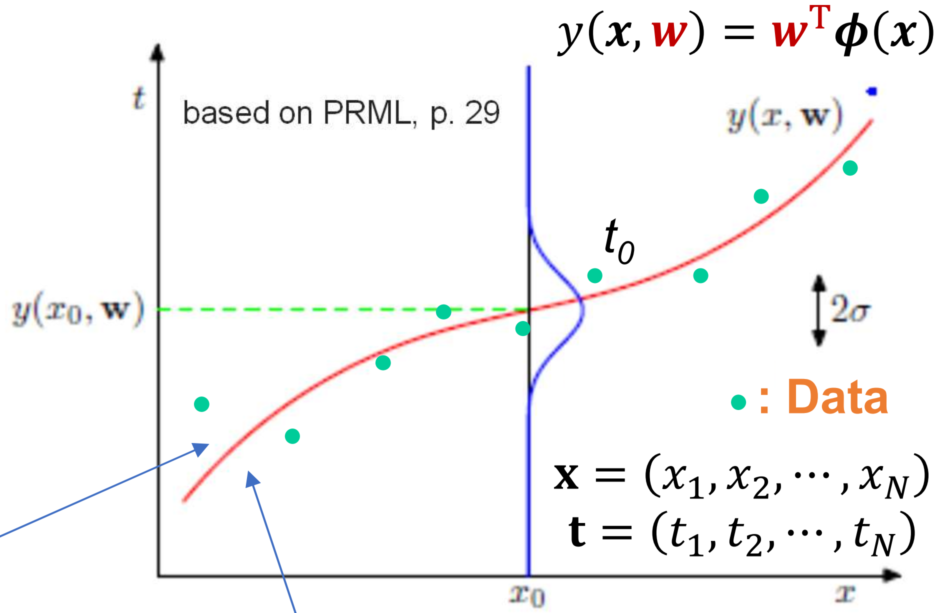
- Nonlinear dimensionality reduction / Other methods

Extension to Nonlinear Dimensionality Reduction

PCA/POD is a **linear** dimensionality reduction method.

We introduced the PCA/POD from this:





in the previous slides.

We know the left figure's case is one special (simple) case of the generalized **linear regression models**.

Manifold*

Kernel PCA (Extension using the Linear Regression Models)

In PCA/POD, we considered the original data $\mathbf{X}$:

$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^{\mathrm{T}}$$

We first map $\boldsymbol{x}$ into a feature space $\boldsymbol{\phi}(\boldsymbol{x})$:

(exactly what we have been doing to consider the linear regression models since lecture 4)

Then, the covariance matrix is described in this situation:

$$\mathbf{S} = \frac{1}{N}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathrm{T}}$$

In the process of solving the eigenvalue problem, the terms including $\boldsymbol{\phi}(\boldsymbol{x})$ are always expressed by $\boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}')$.

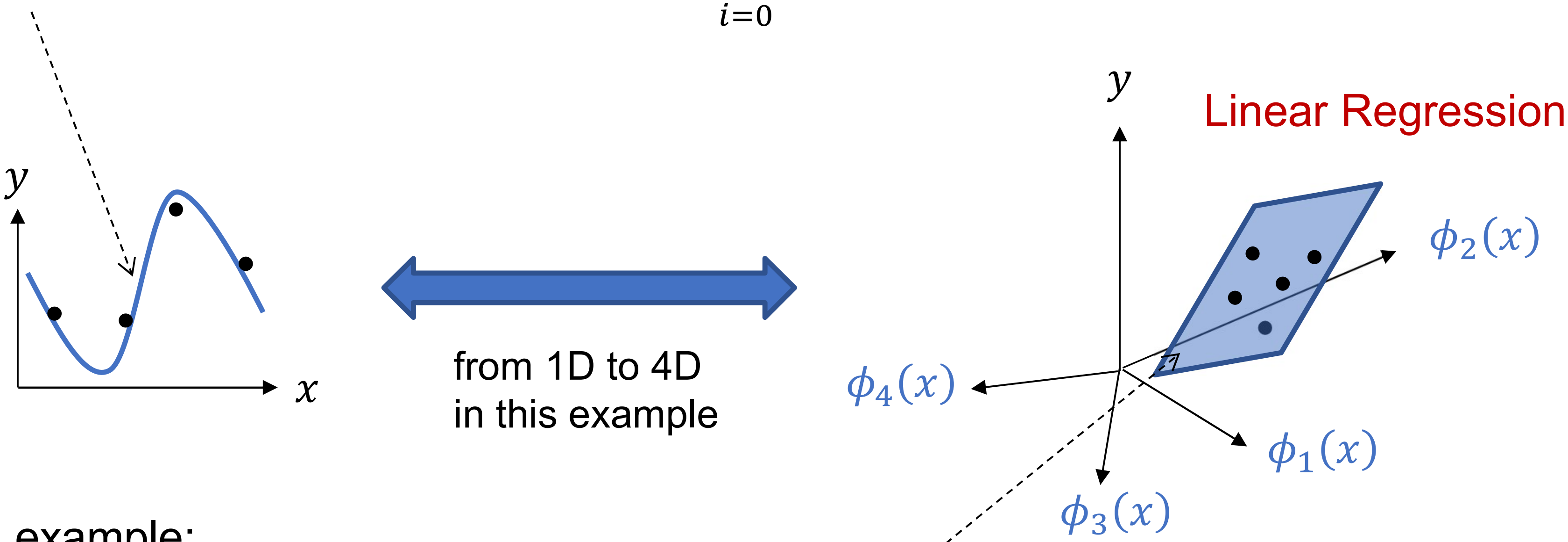
$$k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}')$$

The same as when the kernel trick in Lecture 7 was introduced.

Linear Regression (Repost from Lecture 4)

Shifting to the simplest linear regression by mapping $\boldsymbol{\phi}$: $\boldsymbol{x} \rightarrow \boldsymbol{s}$ $\qquad \boldsymbol{s} = \boldsymbol{\phi}(\boldsymbol{x})$

$$y(x, \boldsymbol{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 = \sum_{i=0}^{3} w_i x^i$$



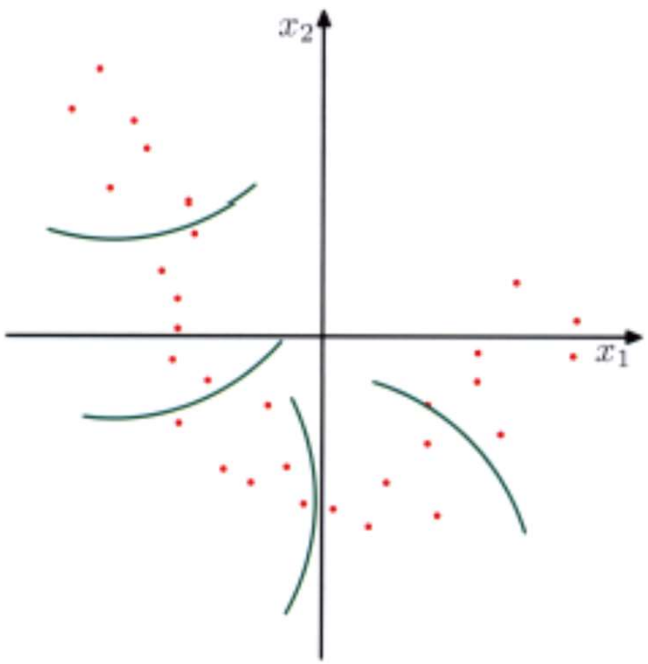
In this example:

$$\boldsymbol{\phi}(x) = \left(\phi_0(x), \phi_1(x), \phi_2(x), \phi_3(x)\right)^{\mathrm{T}}$$
$$= (x^0, x^1, x^2, x^3)^{\mathrm{T}}$$

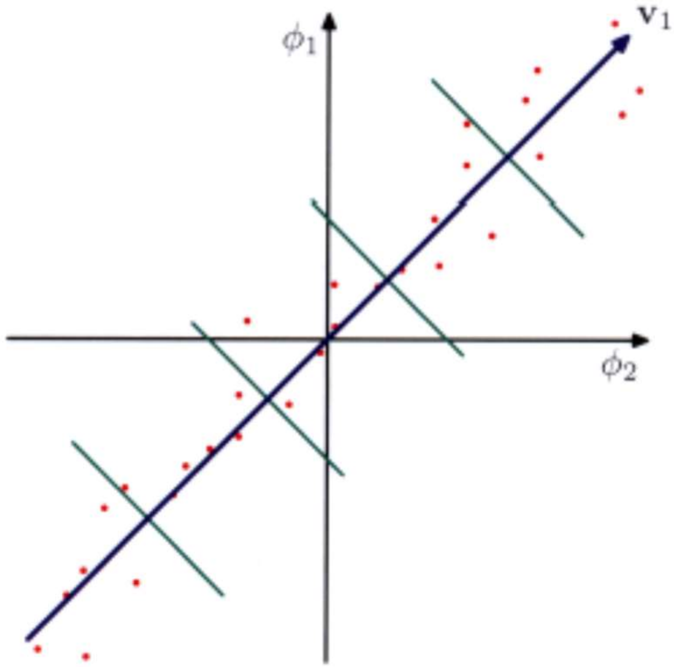
$$y(\boldsymbol{s}, \boldsymbol{a}) = a_0 s_0 + a s_1 + a_2 s_2 + a_3 s_3 = \sum_{i=0}^{3} a_i s_i$$

Kernel PCA

PRML, Fig. 12.16



The original data space (x_1, x_2)

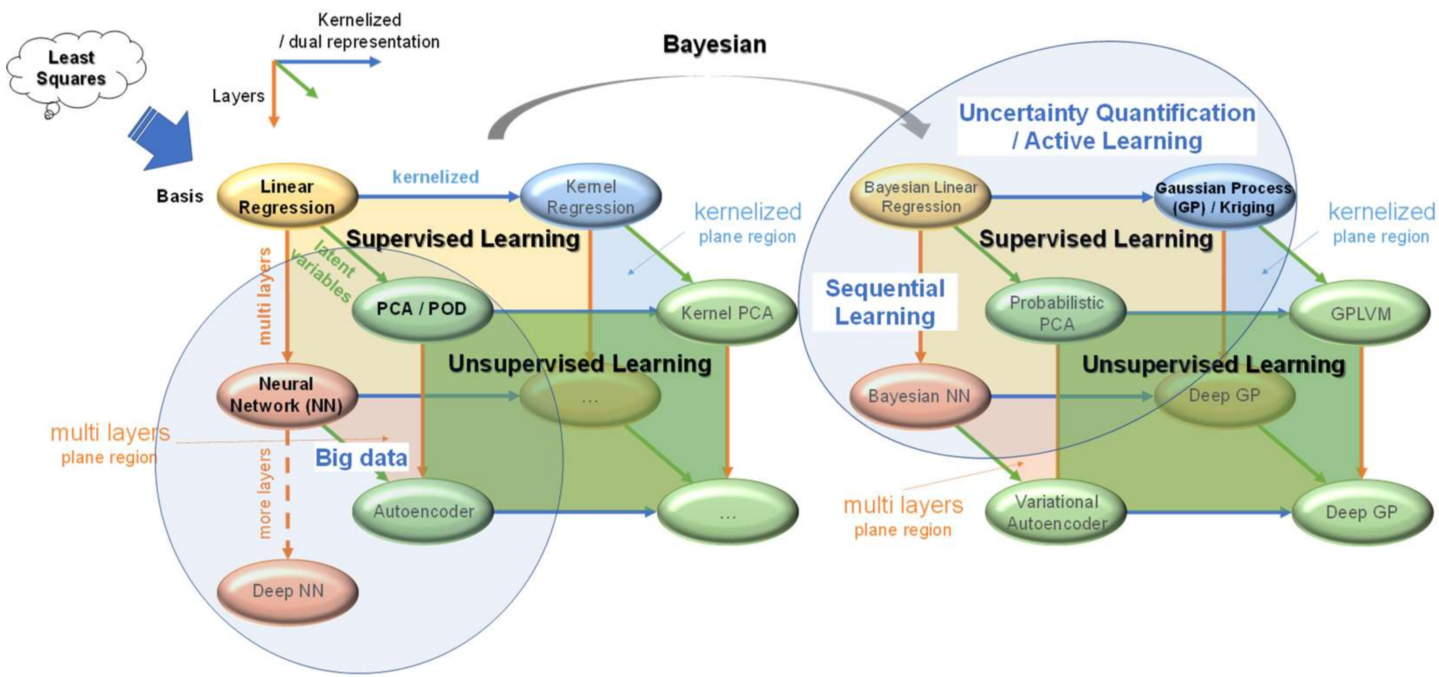


A feature space $(\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}))$

Extension to Nonlinear Dimensionality Reduction

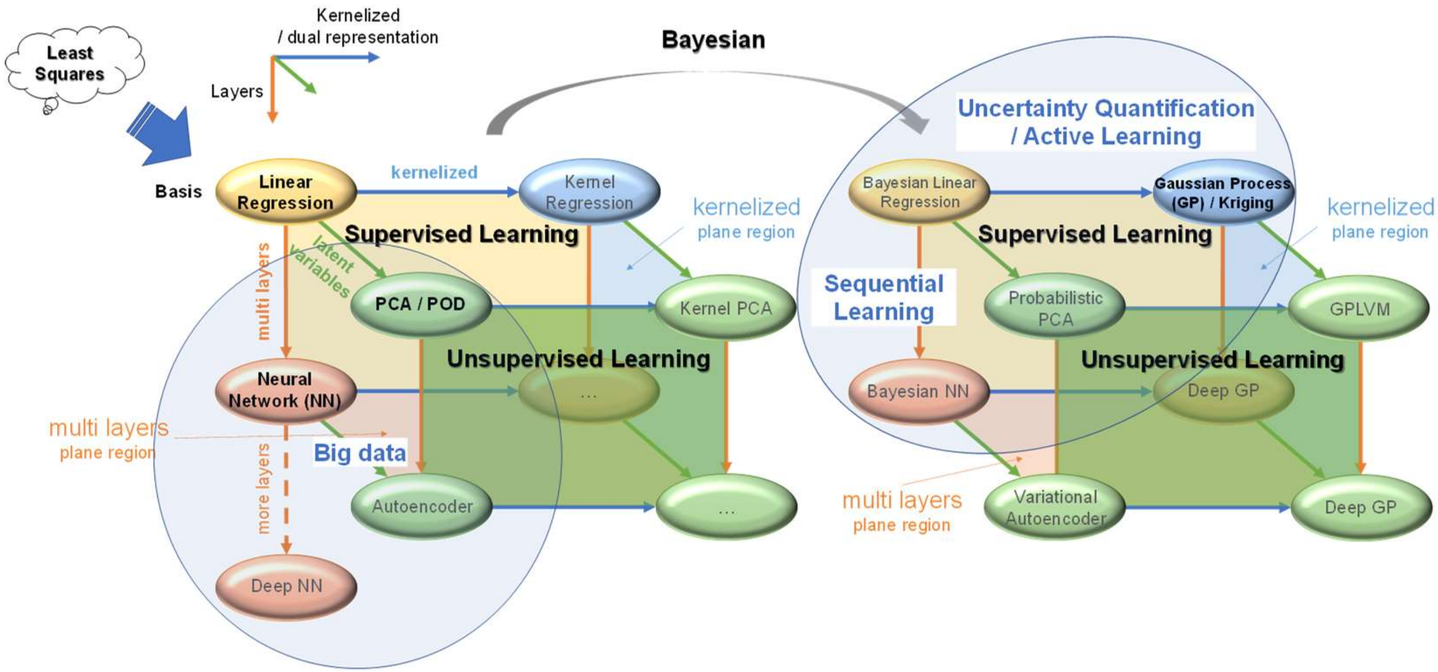
Eventually you may think all the **supervised learning techniques** learned so far can be **extended to unsupervised learning techniques**.

Caution:
Simple linear regression PCA/POD



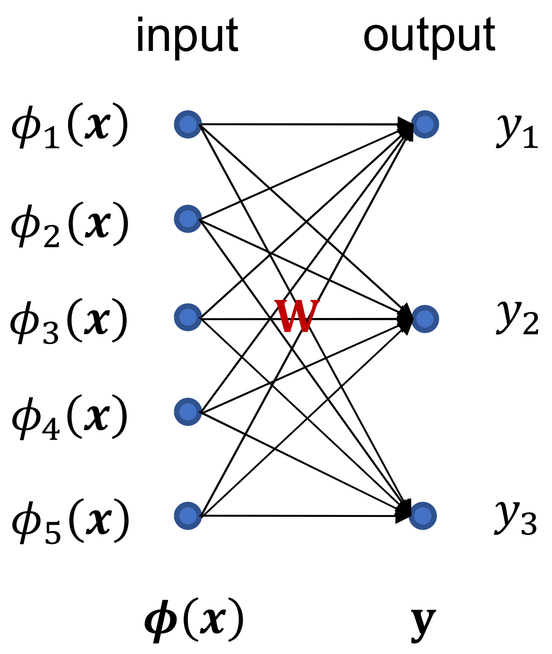
Extension to Nonlinear Dimensionality Reduction

supervised		**unsupervised**
Linear regression		Kernel PCA
Neural network	⇨	Autoencoder
Gaussian process		GPLVM



Linear Regression (multiple output) - Repost from Lecture 10

$$\mathbf{y}(\boldsymbol{x}, \mathbf{W}) = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x})$$



Looks like a neural network?

The tuned parameter $\widehat{\boldsymbol{w}}_i$ (i=1,2,3) can be obtained at the same time as $\widehat{\mathbf{W}}$.

$$\widehat{\mathbf{W}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{T}$$

$$= \begin{pmatrix} & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & \end{pmatrix} \begin{pmatrix} {t_1}^{(1)} & \cdots & {t_3}^{(1)} \\ \vdots & \ddots & \vdots \\ {t_1}^{(N)} & \cdots & {t_3}^{(N)} \end{pmatrix} = \begin{pmatrix} \widehat{w}_{11} & \cdots & \widehat{w}_{13} \\ \vdots & \ddots & \vdots \\ \widehat{w}_{51} & \cdots & \widehat{w}_{53} \end{pmatrix}$$

$$\widehat{\mathbf{W}} = (\widehat{\boldsymbol{w}}_1, \widehat{\boldsymbol{w}}_2, \widehat{\boldsymbol{w}}_3)$$

Consider a multiple output case:

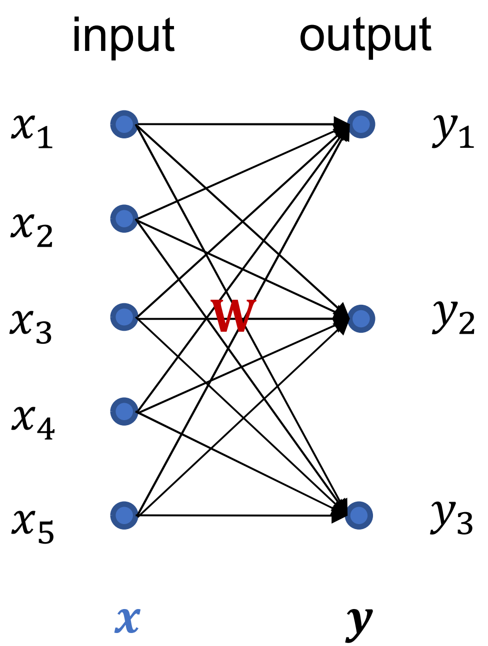
$$\mathbf{T} = \left(\boldsymbol{t}^{(1)}, \boldsymbol{t}^{(2)}, \cdots, \boldsymbol{t}^{(N)}\right)^{\mathrm{T}}$$

This is still the **linear regression model**.

The objective is to obtain $\widehat{\mathbf{W}}$ or $p(\mathbf{W})$ (by using data).

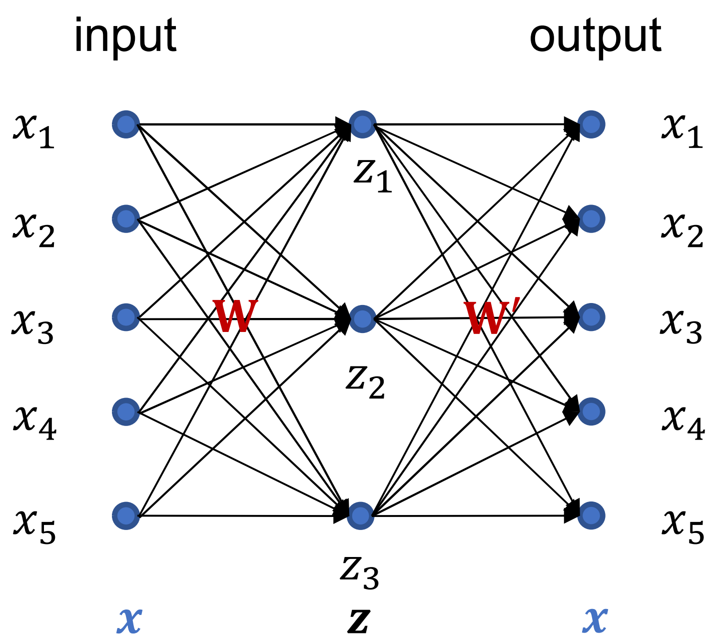
Autoencoder (Extension using the Neural Networks)

(Simple)
Linear regression



This is still the PCA/POD.

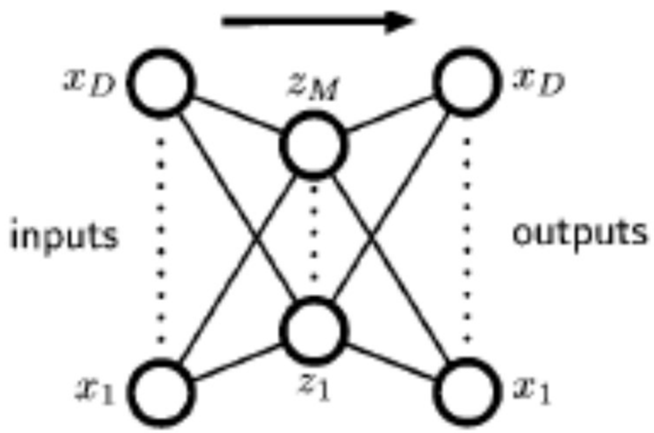
PCA/POD



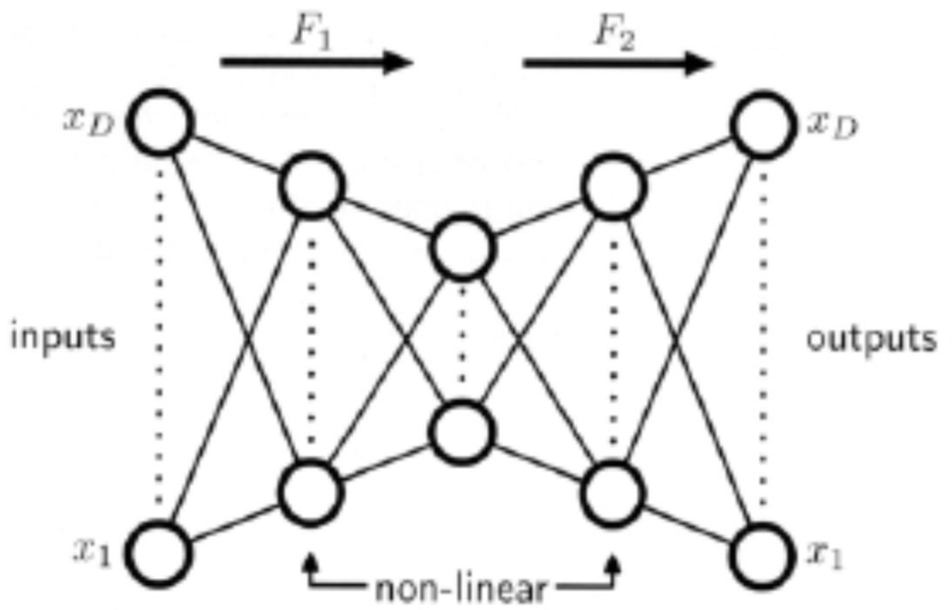
Compare this with $E(\boldsymbol{w})$ of the NN models in Lecture 10.

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} \left\{\boldsymbol{x}^{(n)} - \mathbf{y}\left(\boldsymbol{x}^{(n)}, \boldsymbol{w}\right)\right\}^2$$

Autoencoder (Extension using the Neural Networks)



PRML, Fig. 12.18



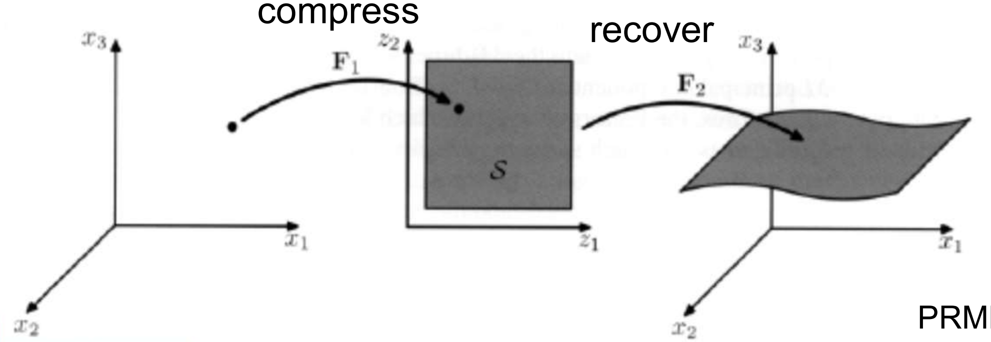
PRML, Fig. 12.19

PCA/POD

Autoencoder

compress

recover



PRML, Fig. 12.20

Combination of the NN and the autoencoder is one of the standard models.

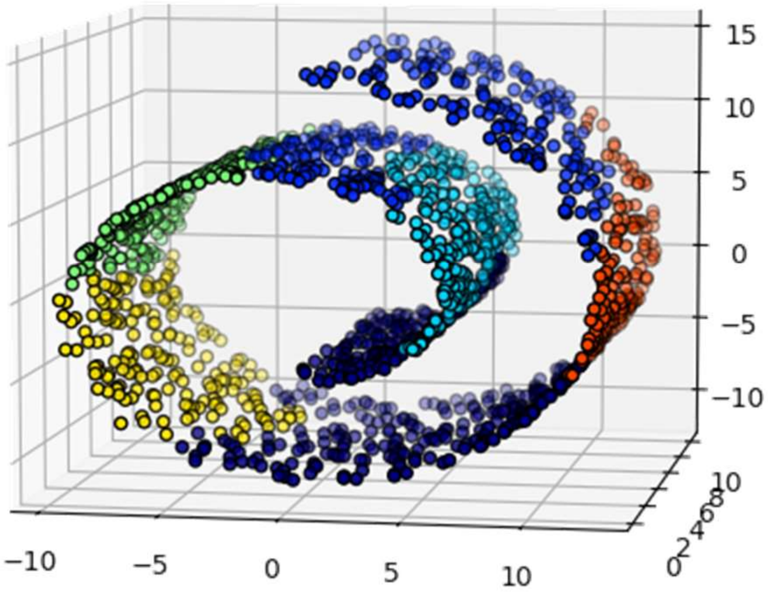
Other Methods

Methods of identifying the **manifold explicitly**

- Locally linear embedding (LLE)
    - smooth joining local linear models

- Isomap
    - assuming an Euclidian space locally
    - Then, take the geodesic (shortest path)

Swiss roll:
benchmark test case



Methods often used in 2D mapping (**visualization**)

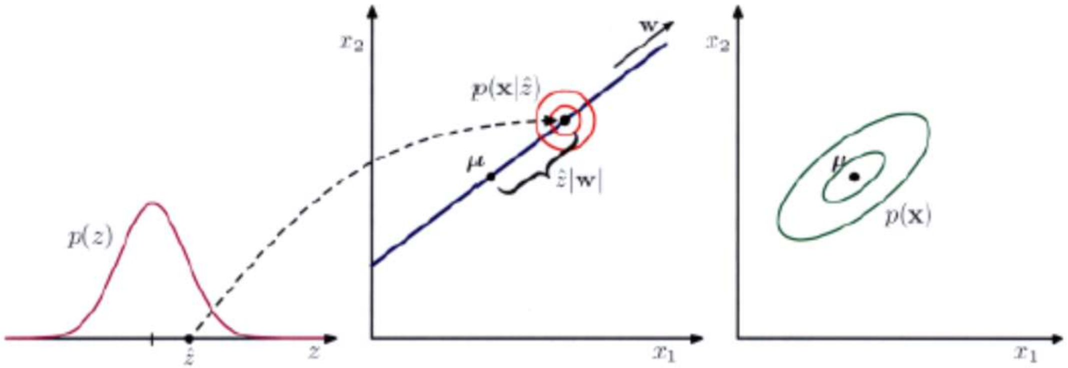
- Self-organizing map (SOM)

- t-SNE

Modelling from the Probabilistic Approaches

The topics here are advanced but related to the topics in Lectures (12) and 13.

This brings the most generalized perspective of modelling.



PRML, Fig. 12.9

Since you may have thought that we started from the least squares, so there might exist a (Gaussian) probabilistic model behind, and also extended to Bayesian approaches.

Probabilistic PCA, Bayesian PCA

This is true. The topics are summarized with other techniques.

The keyword is the Latent variables.

Summary

Unsupervised learning methods were introduced.

- The concepts / applications :
  - Dimensionality reduction
  - Feature extraction
  - 2D, 3D Visualization

- The linear dimensionality reduction (PCA/POD) comes down to the eigenvalue problem.

- Nonlinear dimensionality reduction methods can be naturally extended by the use of the supervised learning techniques learned until here.
  - by regarding the PCA/POD as a simple linear regression model