Deep Learning

A course about theory & practice

Auto-Encoders

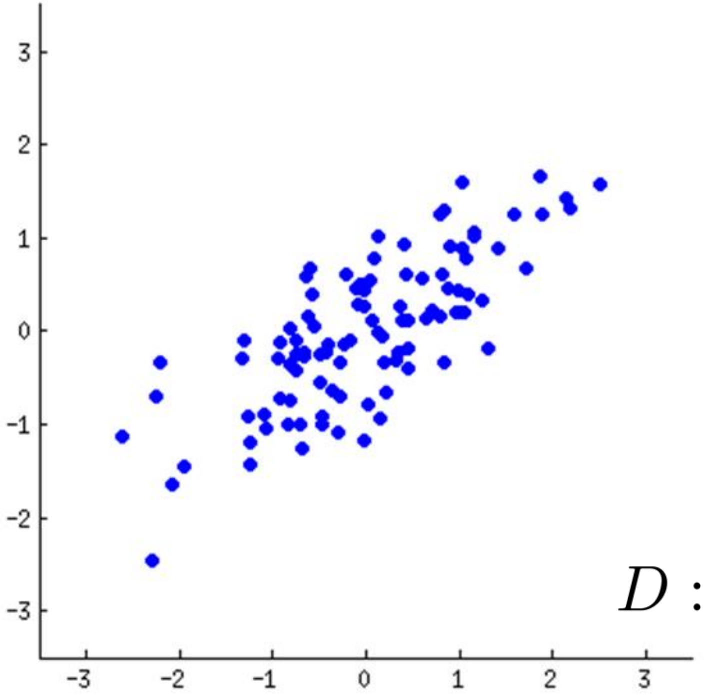
Marco Piastra

Principal Component Analysis (PCA)

PCA: the intuitive idea

- **Dataset of vectors**

  Suppose you have a dataset made of vectors



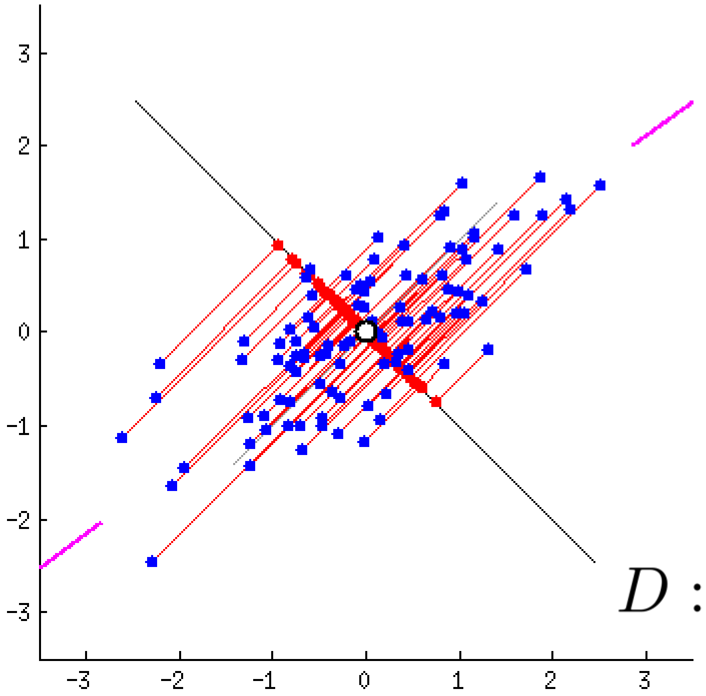
$$D := \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}, \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^d$$

[images from https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues]

PCA: the intuitive idea

- **Change of Basis**

  Suppose you have a dataset made of vectors
  May a change in coordinates, which leaves data unaltered, be advantageous?



$$D := \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}, \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^d$$

[images from https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues]

PCA: the mathematics

- **Translation**

$$D := \{\boldsymbol{x}^{(i)}\}_{i=1}^{N}, \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^d$$

dataset in matrix form

$$\boldsymbol{X} := \begin{bmatrix} x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(N)} & \dots & x_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times d}$$

centroid (mean vector)

$$\boldsymbol{\mu} := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{x}^{(i)}$$

$$\boldsymbol{v}^{(i)} := \boldsymbol{x}^{(i)} - \boldsymbol{\mu}$$

translated dataset in matrix form

$$\boldsymbol{V} := \begin{bmatrix} v_1^{(1)} & \dots & v_d^{(1)} \\ \vdots & \ddots & \vdots \\ v_1^{(N)} & \dots & v_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times d}$$

(it has zero vector mean)

PCA: the mathematics

- **Covariance matrix**

$$\boldsymbol{V} := \begin{bmatrix} v_1^{(1)} & \dots & v_d^{(1)} \\ \vdots & \ddots & \vdots \\ v_1^{(N)} & \dots & v_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times d}$$

Definition of covariance matrix (see Wikipedia)

$$\boldsymbol{C} := \mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^T\right] = \mathbb{E}\left[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\right]$$

$$\boldsymbol{C} = \frac{1}{N}\,\boldsymbol{V}^T\boldsymbol{V}, \qquad \boldsymbol{C} \in \mathbb{R}^{d \times d}$$

Actually, with Bessel correction, this would be $\frac{1}{N-1}$

(it is completely irrelevant here)

PCA: the mathematics

- **Spectral Theorem**

  (a.k.a. Eigenvalue Decomposition – EVD)

  Any square and full-rank matrix like:

$$\boldsymbol{C} = \boldsymbol{V}^T \boldsymbol{V}, \quad \boldsymbol{C} \in \mathbb{R}^{d \times d}$$

  can be decomposed as

$$\boldsymbol{C} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T$$

  where $\boldsymbol{U}$ is *orthogonal* and $\boldsymbol{\Lambda}$ is *diagonal*

$$\boldsymbol{\Lambda} := \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_d \end{bmatrix} \qquad \boldsymbol{\lambda} := [\lambda_1, \dots, \lambda_d]$$

$\boldsymbol{U}$ *Eigenvectors: the new vector basis*

$\boldsymbol{\lambda}$ *Eigenvalues: multipliers, or 'the mass' of the original matrix*

  Furthermore, any covariance matrix is *semidefinite positive*, which means

$$\lambda_i \geq 0, \forall i \in \{1, \dots, d\}$$

PCA: the mathematics

- **Dimension reduction**

  Rotation matrix

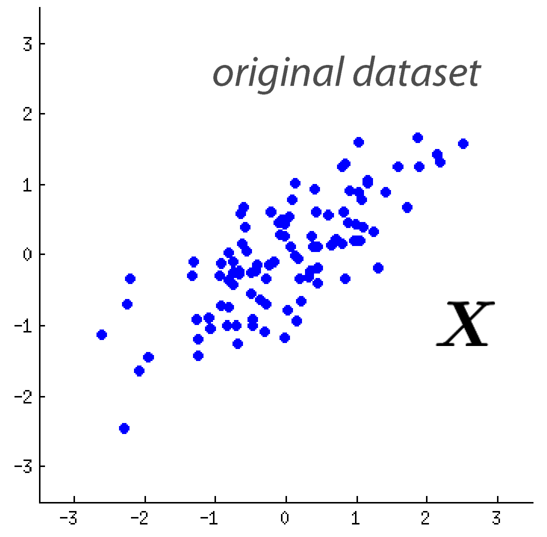
$$\boldsymbol{U} \in \mathbb{R}^{d \times d}$$

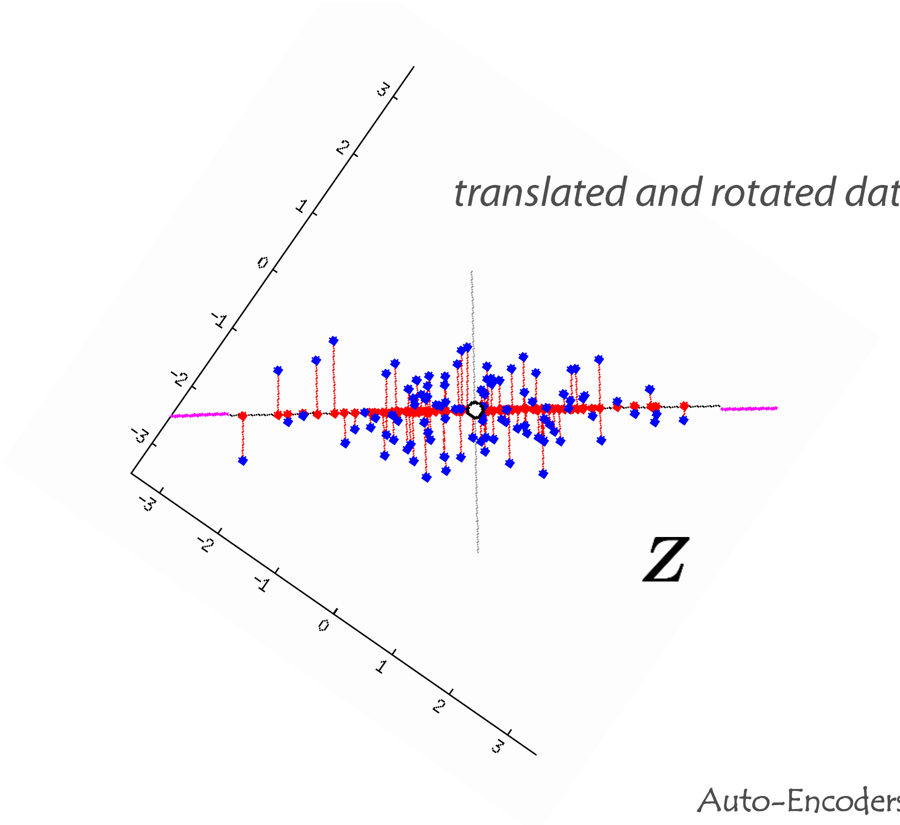
— *Eigenvectors: each row is a versor for the new coordinate space*

  Projecting data onto the new feature space

$$\boldsymbol{Z} := \boldsymbol{V}\boldsymbol{U}^T \in \mathbb{R}^{N \times d}$$

  Since $\boldsymbol{U}$ and $\boldsymbol{U}^T$ are orthogonal, the linear transformation is a pure *rotation* around the (translated) origin

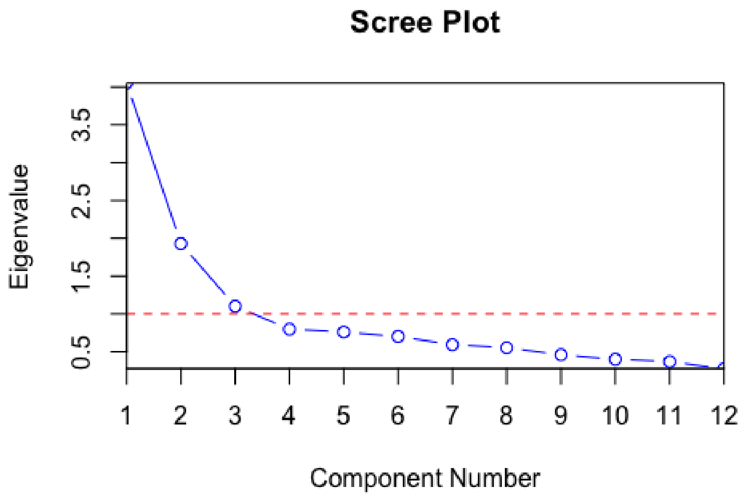




PCA: what is this all for

- **Dimension reduction**

  Sorting eigenvalues in $\boldsymbol{\lambda}$ (scree plot)



[image from https://en.wikipedia.org/wiki/Scree_plot]

  Selecting principal components

$$\hat{\boldsymbol{\lambda}} \in \mathbb{R}^r, \quad r < d$$

  Projection matrix

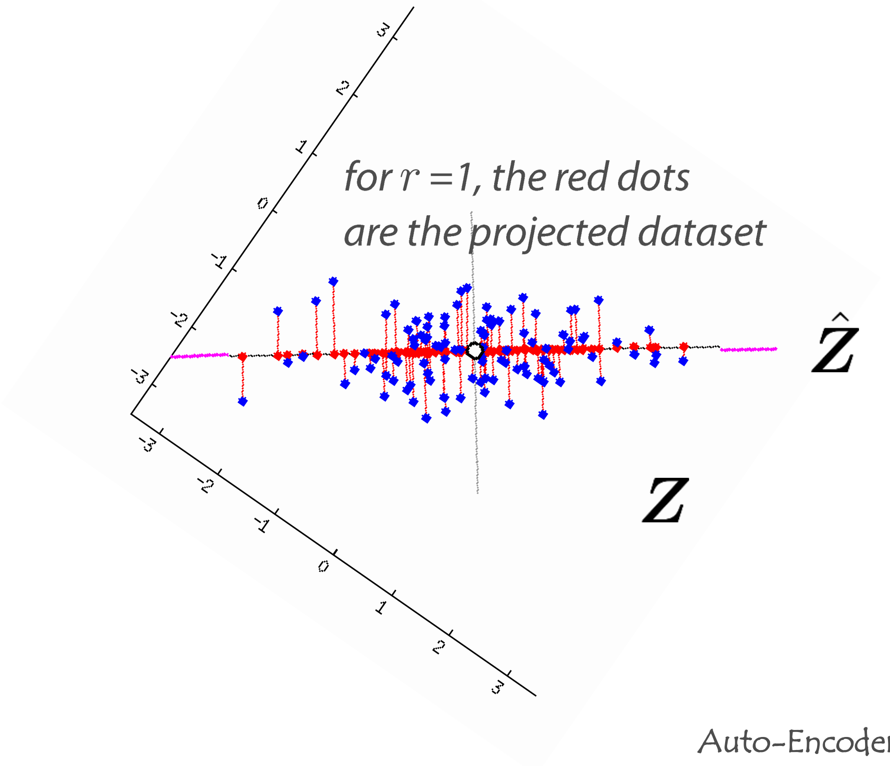
$$\boldsymbol{U} \in \mathbb{R}^{d \times d}$$

$$\hat{\boldsymbol{U}} \in \mathbb{R}^{d \times r}$$

only the selected r columns have been preserved

  Projecting data onto the new feature space

$$\hat{\boldsymbol{Z}} := \boldsymbol{V}\hat{\boldsymbol{U}}^T \in \mathbb{R}^{N \times r}$$



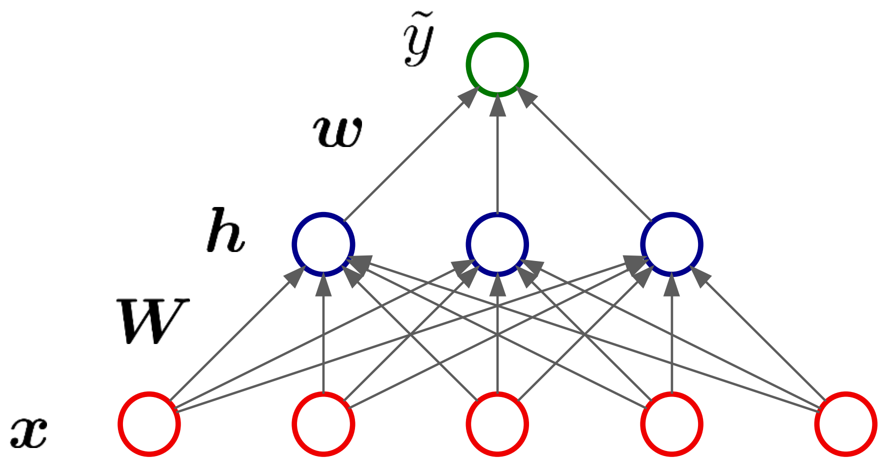
Auto-Encoders

Auto-Encoders

- **Encoder**

  A feed-forward neural network with one hidden layer

$$\tilde{y} = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$



Auto-Encoders

- **Encoder**

  A feed-forward neural network with one hidden layer

  $$\tilde{y} = \boldsymbol{w} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + b$$

- **Auto-encoder** (*basic idea*)**: encoder + decoder**

  $$\boldsymbol{x}^{[m]} = g(\boldsymbol{W}^{[m]} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}^{[m]})$$

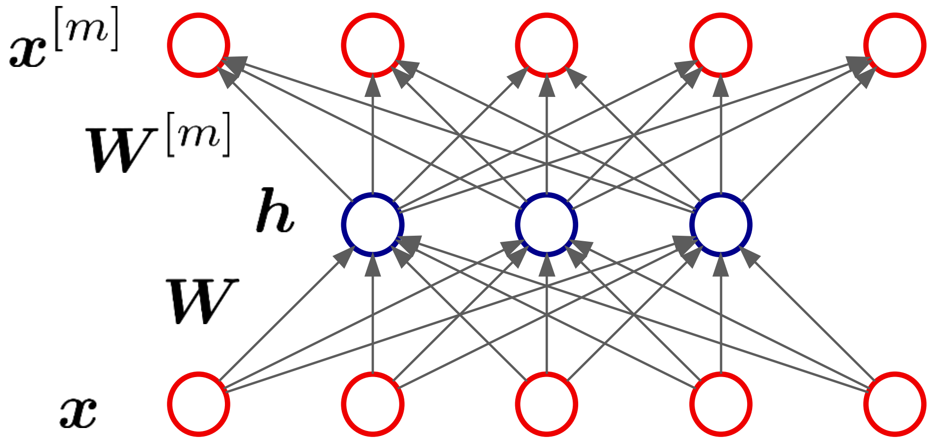
  Loss function (MSE):

  $$L(\boldsymbol{x}^{[m]}, \boldsymbol{x}) = (\boldsymbol{x}^{[m]} - \boldsymbol{x})^2$$

  Initially:

  $$\boldsymbol{W}^{[m]} = \boldsymbol{W}^T, \ \boldsymbol{W} \in \mathbb{R}^{r \times d}$$

  then train the network with each data sample ***onto itself*** *(unsupervised learning)*



Linear Auto-Encoders (vs PCA)

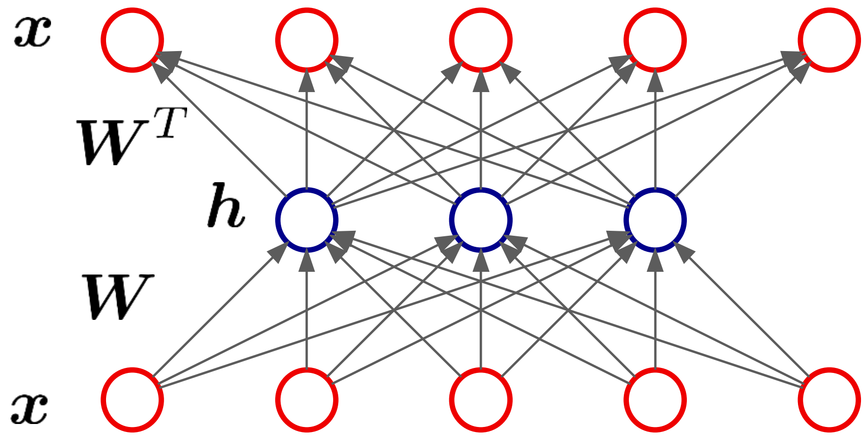
- **Linear Auto-encoder** (*basic idea*)

$$\tilde{\boldsymbol{x}} = (\boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}')$$

Look ma': no non-linear, activation functions

$$\tilde{\boldsymbol{v}} = \boldsymbol{W}^T\boldsymbol{W}\boldsymbol{v}$$

Use translated dataset vectors and set vector bias to zero



The loss function aims to achieve $\tilde{\boldsymbol{v}} - \boldsymbol{v} = \boldsymbol{0}$
therefore, it can be rewritten as:

$$\left\| \boldsymbol{V}\boldsymbol{W}^T\boldsymbol{W} - \boldsymbol{V} \right\|_F^2$$

Frobenius norm of a matrix: (flatten it and take the norm)

Mathematically, it can be shown that such loss has a unique global minimum in which the line vectors in $\boldsymbol{W}$
are the r most significant eigenvectors of $\boldsymbol{C} = \frac{1}{N}\boldsymbol{V}^T\boldsymbol{V}$
(up to a scaling factor)

[see https://arxiv.org/pdf/1804.10253]

Deep Auto-Encoders

- **Shallow Auto-encoder**

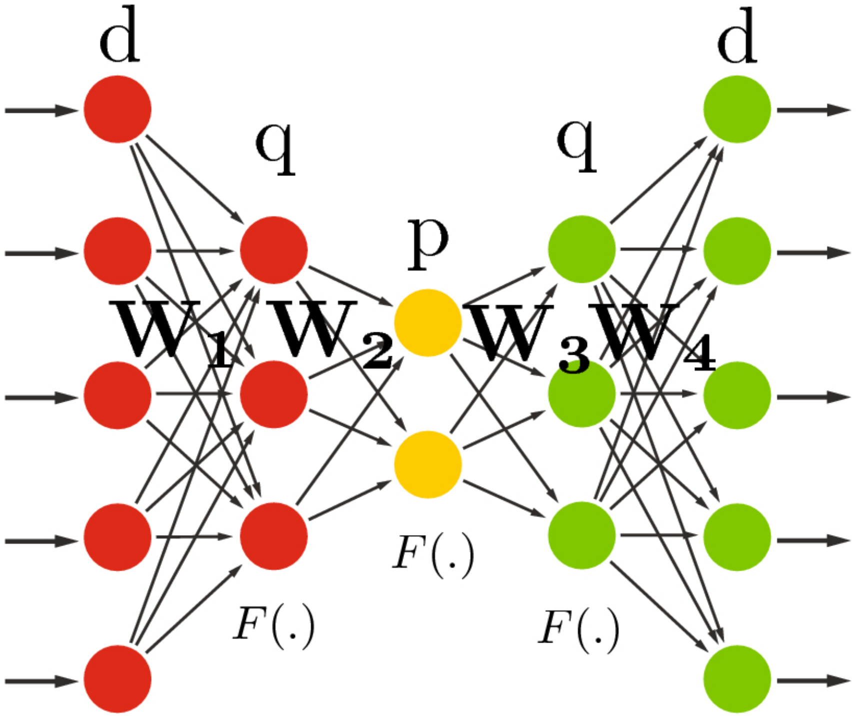
$$\boldsymbol{x}^{[m]} = g(\boldsymbol{W}^{[m]} \cdot g(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) + \boldsymbol{b}^{[m]})$$

It can be shown that also with one non-linear layer per each side, the optimum $\boldsymbol{W}$ still relates to the r most significant eigenvectors (PCA)

- **Deep Auto-Encoder**

It takes at least *two* non-linear layers per each side to achieve a truly non-linear auto-encoder

[Bourlard & Kabil, *Autoencoders reloaded*, 2022]



[image from Bourlard & Kabil, 2022 -https://link.springer.com/article/10.1007/s00422-022-00937-6]

Auto-Encoders

- **Auto-encoder** *(More in general)*

  Two main (composite) layers: **encoder** and **decoder**

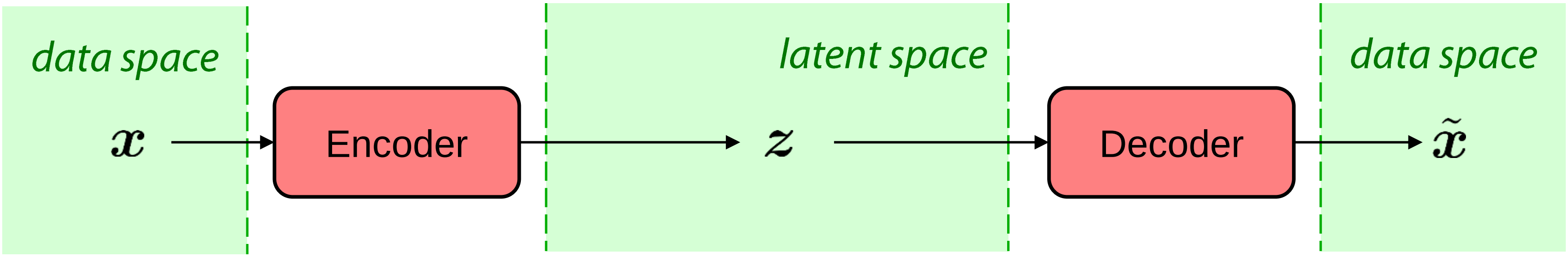
  One **hidden** or **latent** layer $\boldsymbol{z}$

  Each item in the dataset comprises the input only (*Unsupervised Learning*)

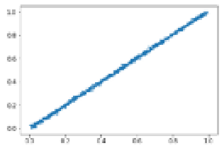
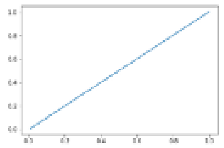
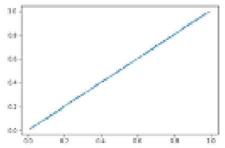
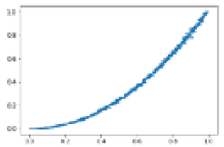
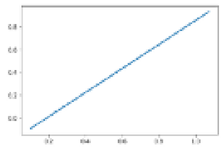
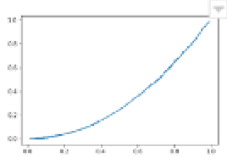
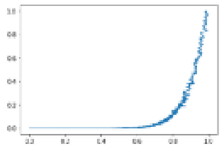
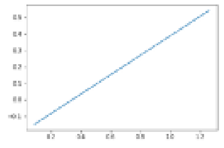
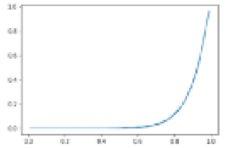
$$D := \{(\boldsymbol{x}^{(i)})\}_{i=1}^{N},$$

  The result of the optimization is $\boldsymbol{z}$:
  a compact (i.e. lower-dimensional) representation of the input $\boldsymbol{x}$

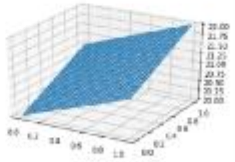
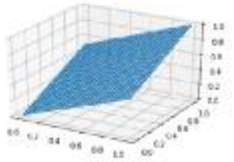
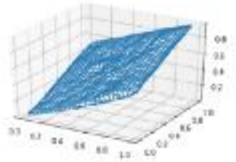
  This representation is also called the *latent space*



Auto-Encoders vs PCA

Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
y=mx+c			
y=mx²+c			
y=mx⁸+c			

When non-linearity matters...

Function	Feature Space	PCA Reconstruction	Auto Encoder Reconstruction
Plane			
Curved Surface