

Deep Learning

A course about theory & practice

Auto-Encoders

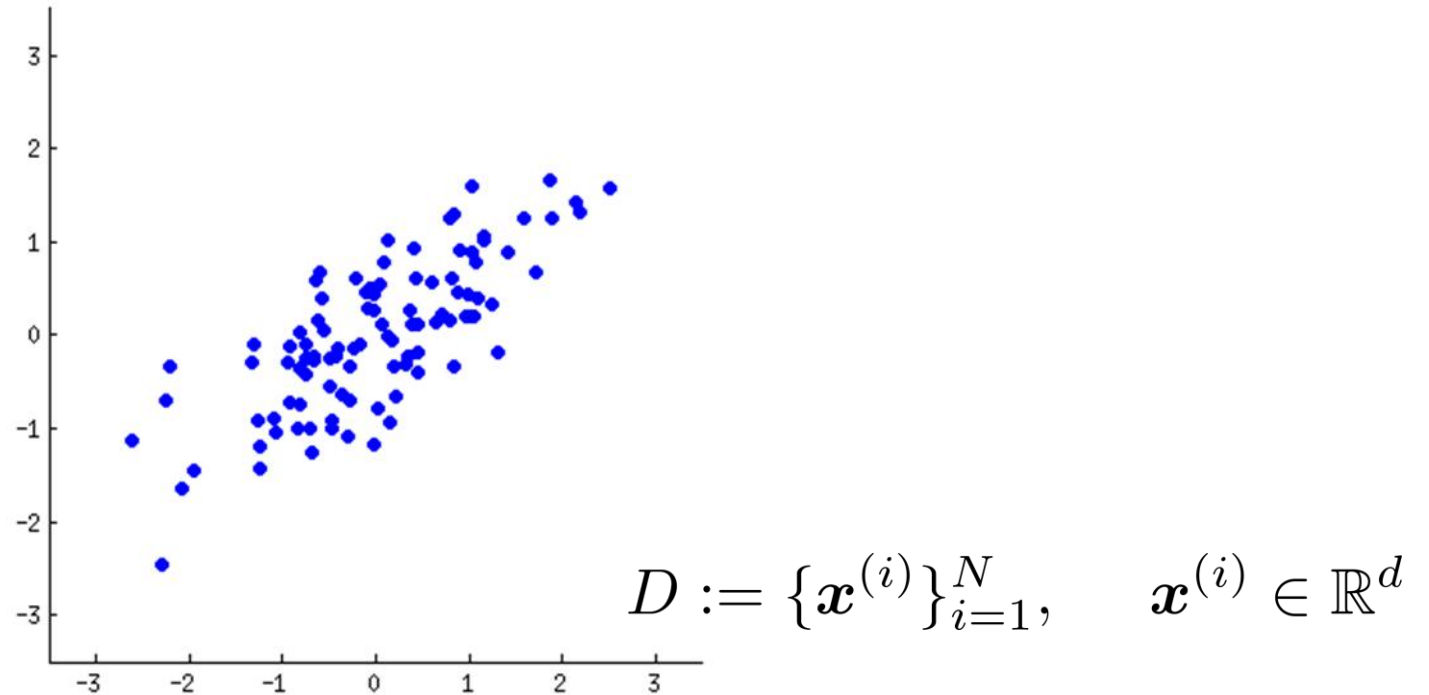
Marco Piastra

Principal Component Analysis (PCA)

PCA: the intuitive idea

■ Dataset of vectors

Suppose you have a dataset made of vectors



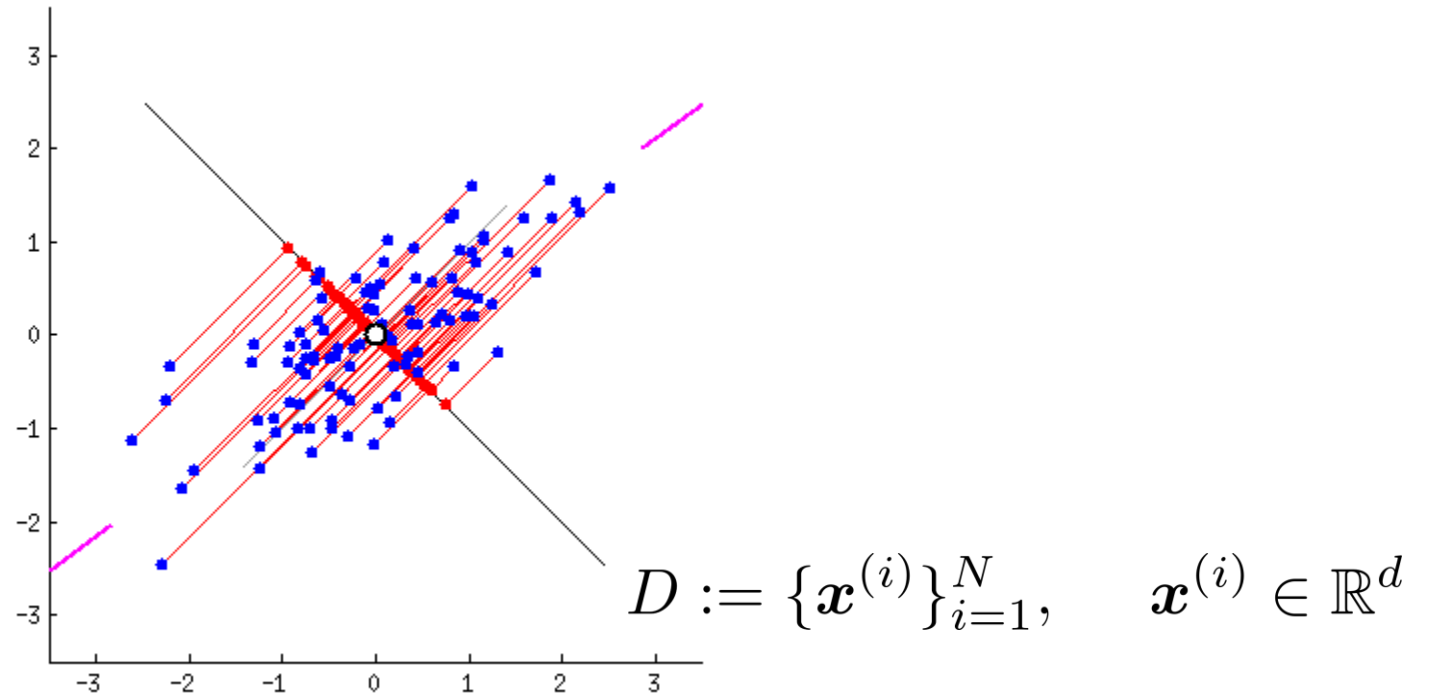
[images from <https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>]

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?



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

PCA: the mathematics

■ Translation

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

dataset in matrix form

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

centroid (mean vector)

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

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

translated dataset
in matrix form

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

(it has zero vector mean)

PCA: the mathematics

■ Covariance matrix

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

Definition of covariance matrix (see Wikipedia)

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

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

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:

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

can be decomposed as

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

where \mathbf{U} is orthogonal and $\mathbf{\Lambda}$ is diagonal

\mathbf{U} Eigenvectors: the new vector basis

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

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

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

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

PCA: the mathematics

■ Dimension reduction

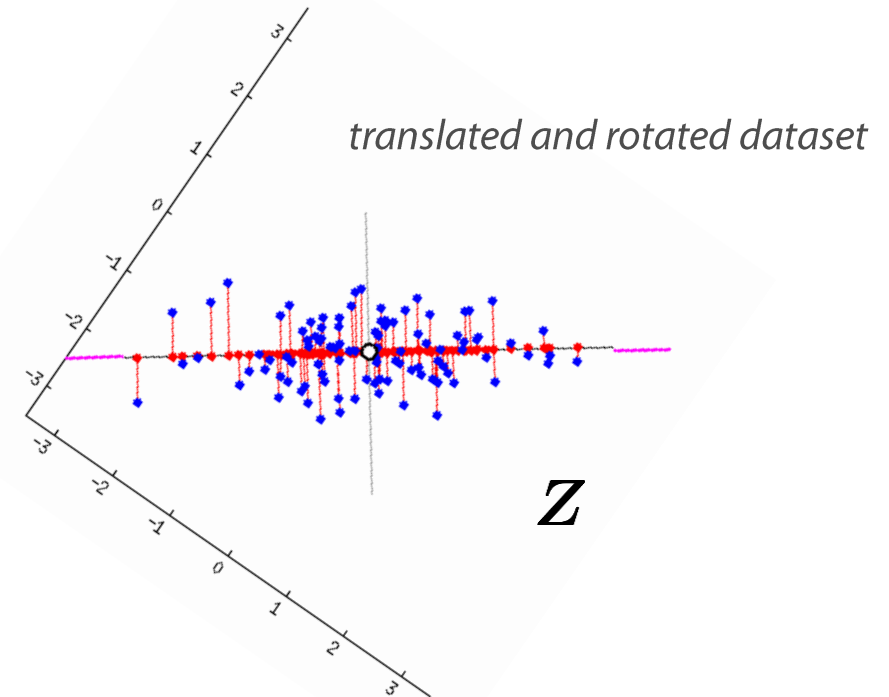
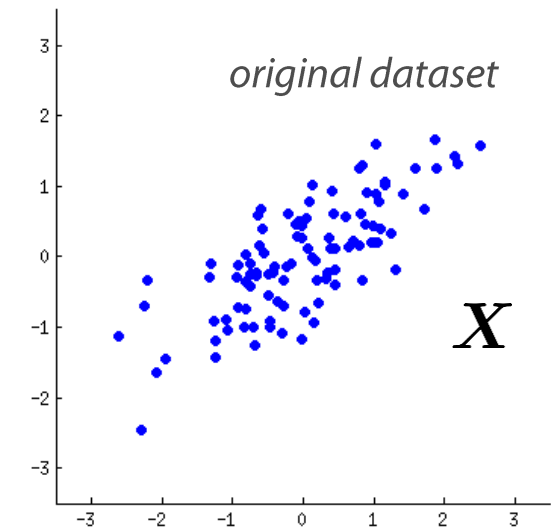
Rotation matrix

$$\mathbf{U} \in \mathbb{R}^{d \times d} \quad \text{—} \quad \textit{Eigenvectors: each row is a versor for the new coordinate space}$$

Projecting data onto the new feature space

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

Since \mathbf{U} and \mathbf{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 λ (scree plot)

Selecting principal components

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

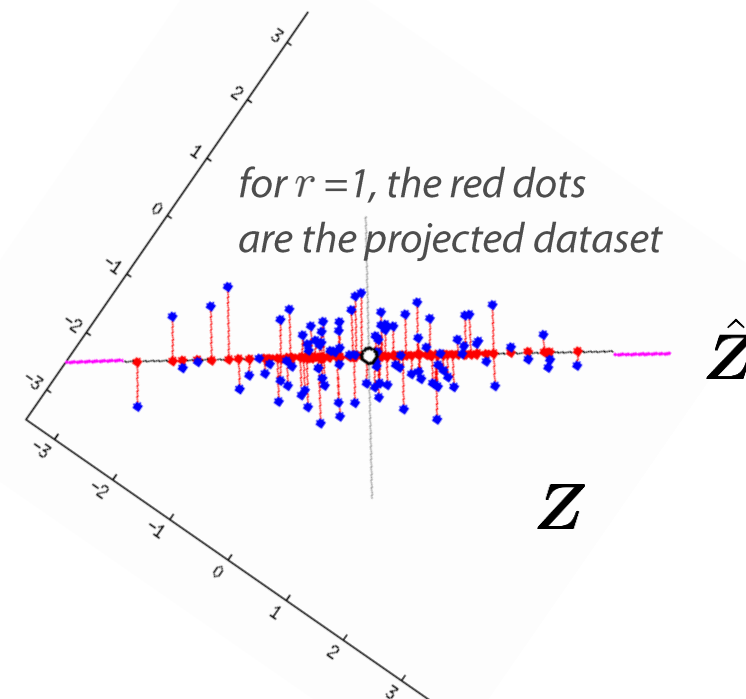
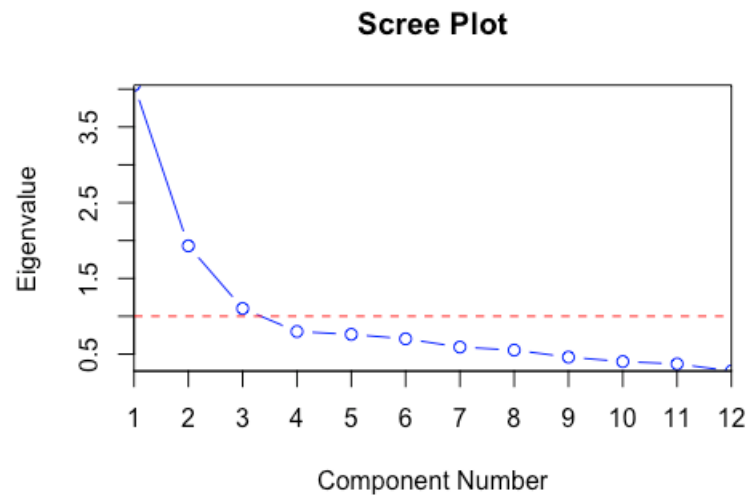
Projection matrix

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

$$\hat{\mathbf{U}} \in \mathbb{R}^{d \times r} \quad \text{--- only the selected } r \text{ columns} \\ \text{have been preserved}$$

Projecting data onto the new feature space

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



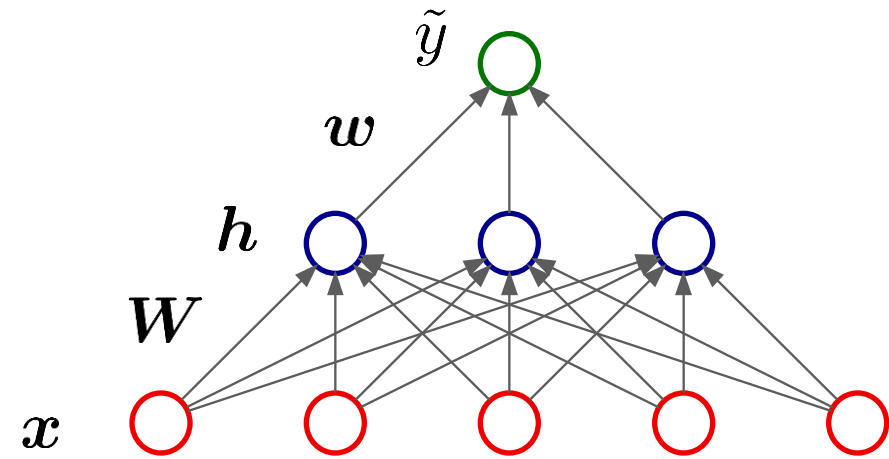
Auto-Encoders

Auto-Encoders

■ Encoder

A feed-forward neural network with one hidden layer

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



Auto-Encoders

■ Encoder

A feed-forward neural network with one hidden layer

$$\tilde{y} = w \cdot g(Wx + b) + b$$

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

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

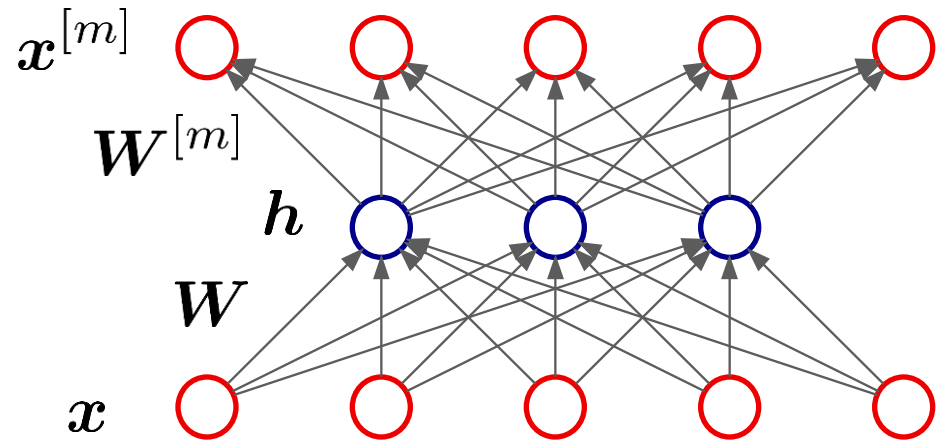
Loss function (MSE):

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

Initially:

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

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



Linear Auto-Encoders (vs PCA)

■ Linear Auto-encoder (basic idea)

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

Notice the absence of non-linear activation functions...

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

Use translated dataset vectors and set vector bias to zero

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

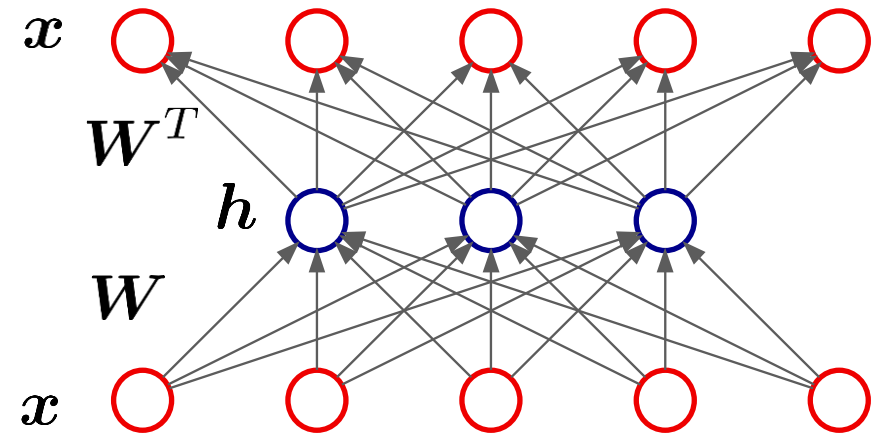
$$\|\mathbf{V}\mathbf{W}^T\mathbf{W} - \mathbf{V}\|_F^2$$

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 \mathbf{W}
are the r most significant eigenvectors of:

$$\mathbf{C} = \frac{1}{N} \mathbf{V}^T \mathbf{V}$$

(up to a scaling factor)



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

Deep Auto-Encoders

■ Shallow Auto-encoder

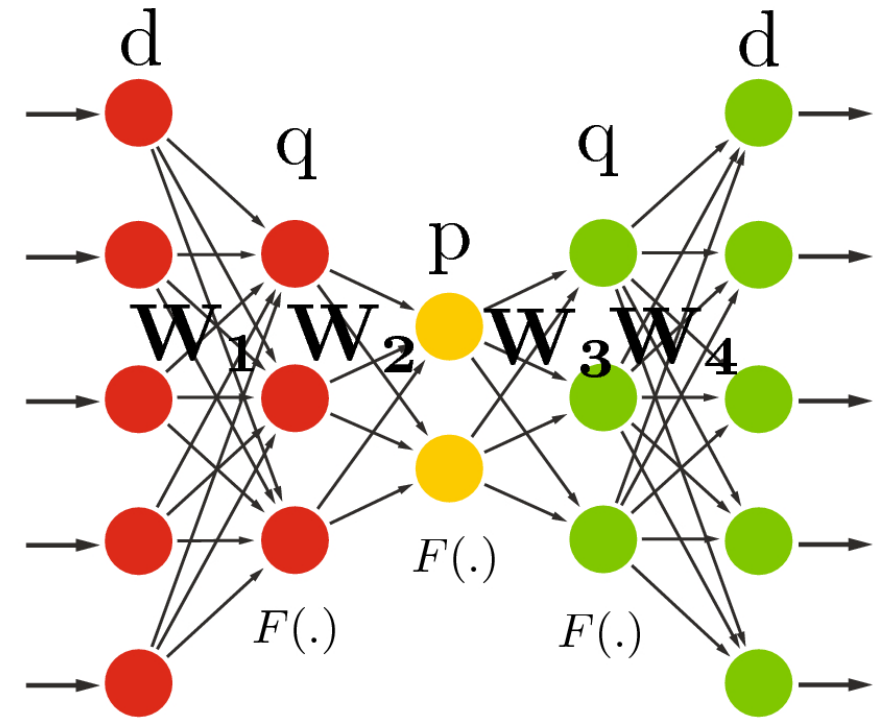
$$\mathbf{x}^{[m]} = g(\mathbf{W}^{[m]} \cdot g(\mathbf{W}\mathbf{x} + \mathbf{b}) + \mathbf{b}^{[m]})$$

It can be shown that also with one non-linear layer per each side, the optimum \mathbf{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 z

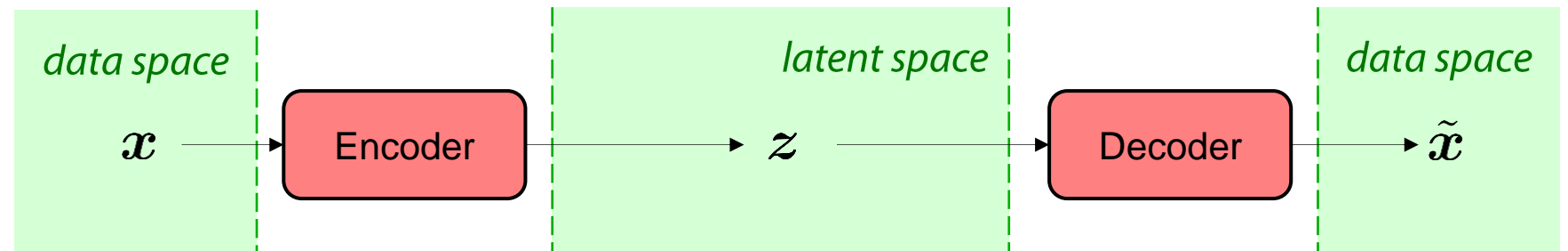
Each item in the dataset comprises the input only (*Unsupervised Learning*)

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

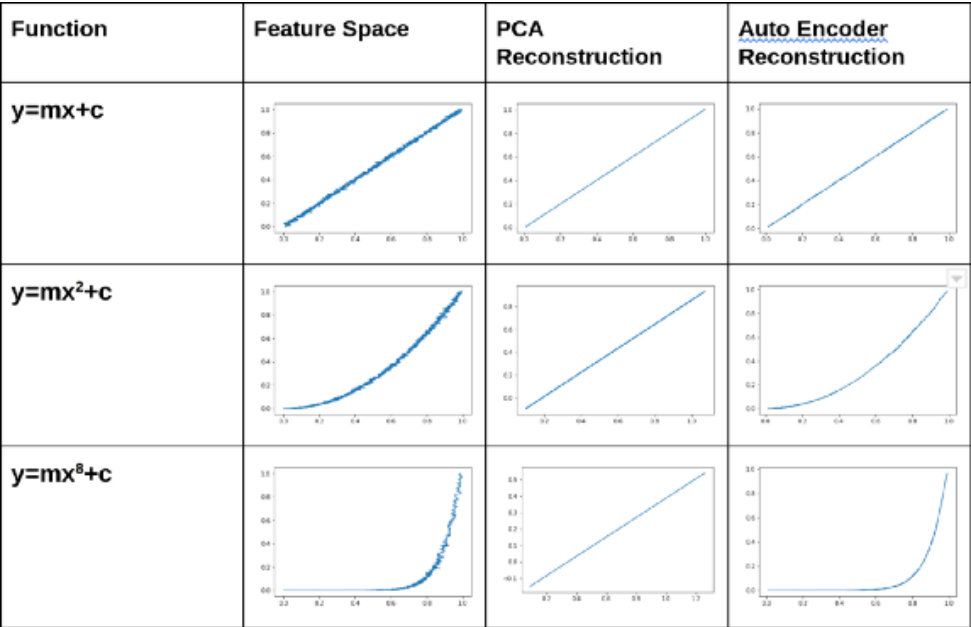
The result of the optimization is z :

a compact (i.e. lower-dimensional) representation of the input \mathbf{x}

This representation
is also called
the *latent space*



Auto-Encoders vs PCA



When non-linearity matters...

