University of Amsterdam
Informatics Institute

AMLAB
Amsterdam Machine Learning Lab

Machine Learning 1

Lecture 10.4 - Unsupervised Learning
Non-linear PCA

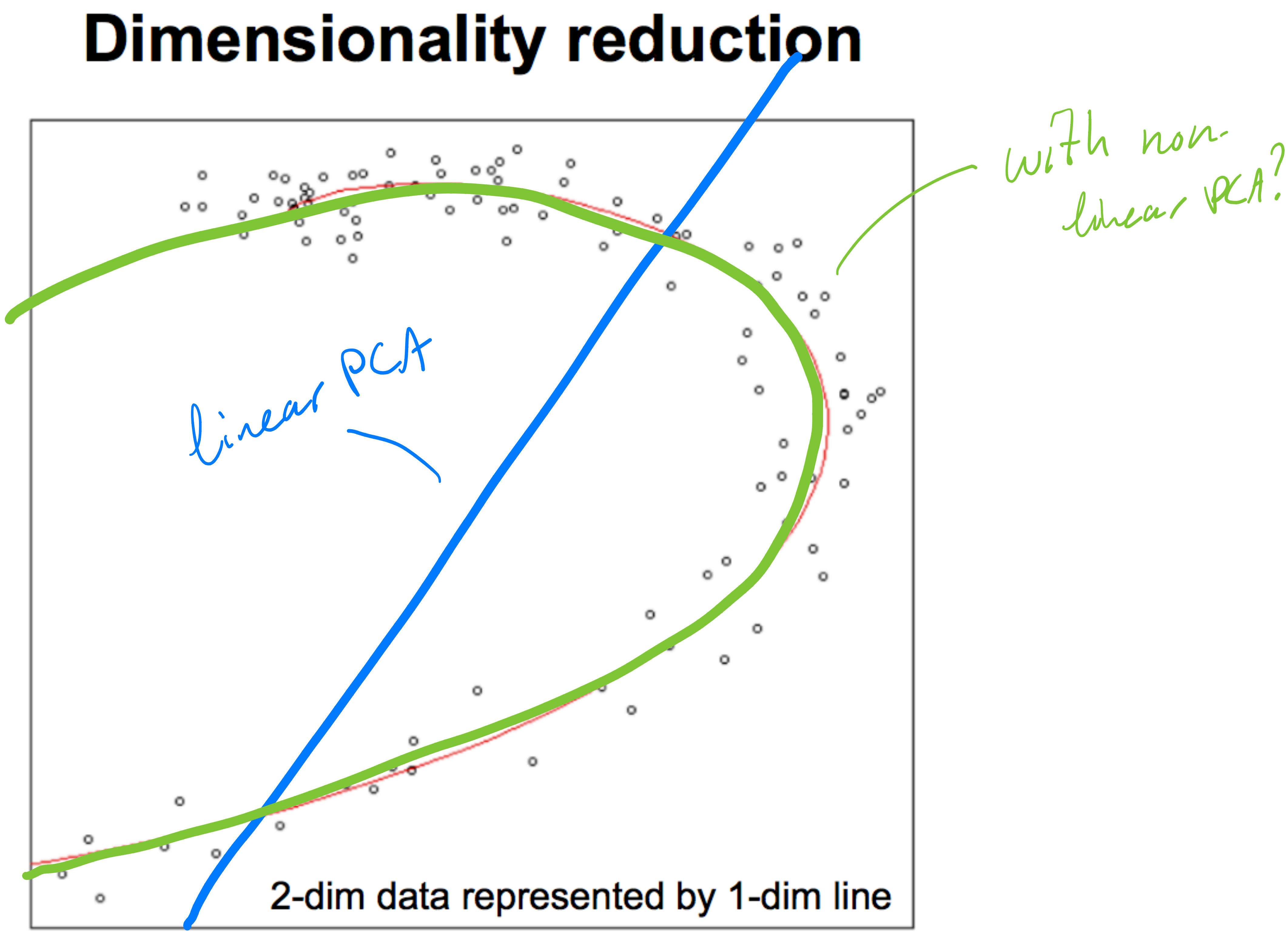
Erik Bekkers

(Bishop 12.3, (12.4.1), 12.4.2)

Slide credits: Thomas Kipf and Giorgio Patrini

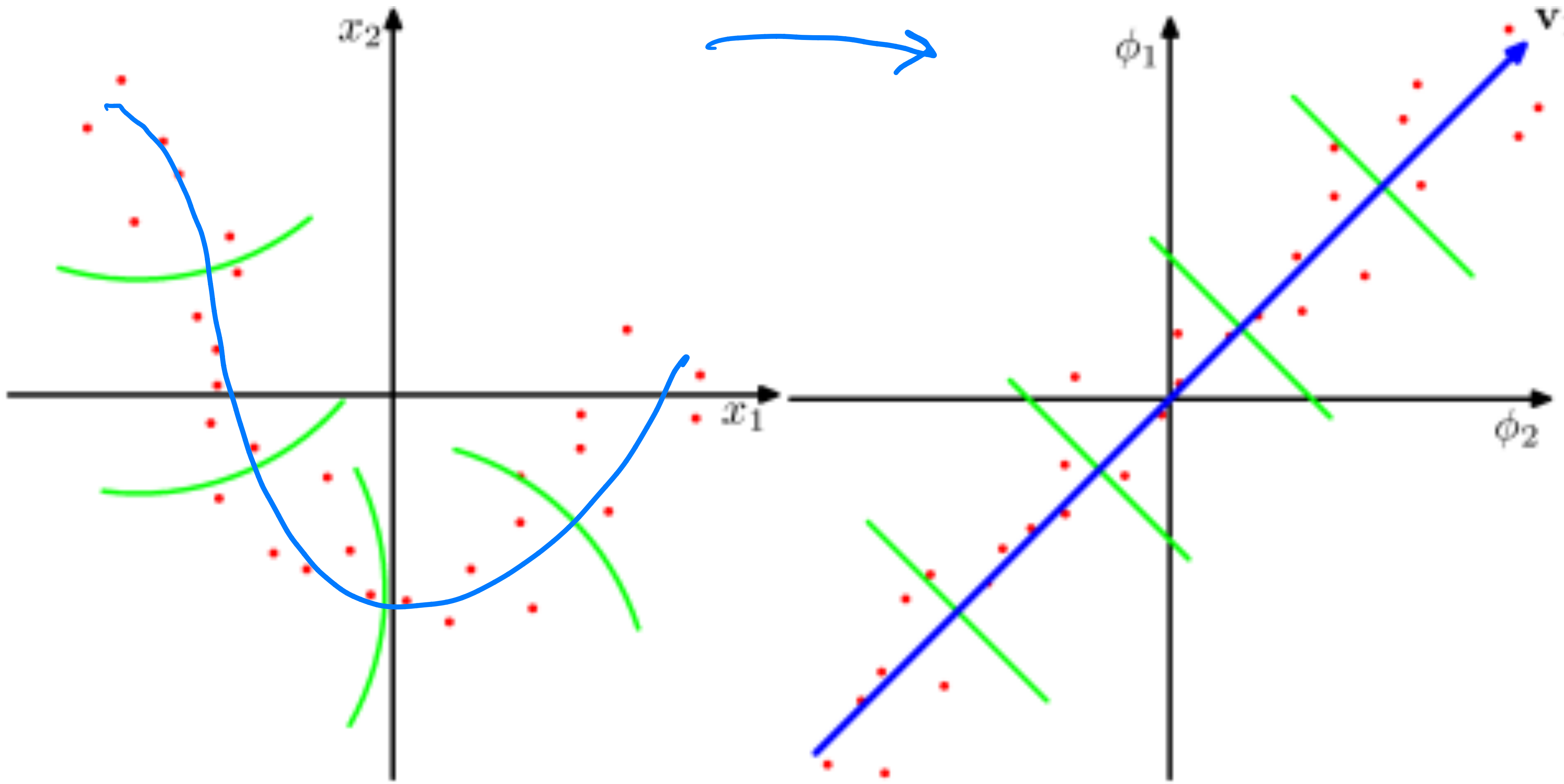
Image credit: Kirillm | Getty Images

PCA on a spiral



PCA using basis functions

use predefined feature vectors $\underline{\phi}(\underline{x})$



(Bishop 12.16)

How to choose $\underline{\phi}$?

Kernel PCA (Bishop 12.3)

$\mathrm{Cov}[x] = \frac{1}{N}\sum_n (x_n - \bar{x})(x_n - \bar{x})^T$

- Use feature transformations to "linearize" the data
  - Via some specific choice of basis functions $\boldsymbol{\phi}(\mathbf{x}_n)$
  - Do PCA in this space

$$S = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^T, \qquad C = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^T$$

$\mathbb{R}^M$

- Kernel approach:
  - Let $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^T\boldsymbol{\phi}(\mathbf{x}_j)$ be the kernel associated with the basis functions
  - Let $\mathbf{u}_i$ be the principal components of C $\in \mathbb{R}^{M \times M}$
  - Let $z_i(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^T\mathbf{u}_i$ the projection onto the i^{th} component

$\mathbb{R}^N$

  - Let $\mathbf{a}_i$ be the i^{th} eigenvector of $\mathbf{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$ $\in \mathbb{R}^{N \times N}$
  - Then $z_i(\mathbf{x}) = \sum_{n=1}^{N} a_{in} k(\mathbf{x}, \mathbf{x}_n)$ - The projection is purely in terms of the other data points via k!

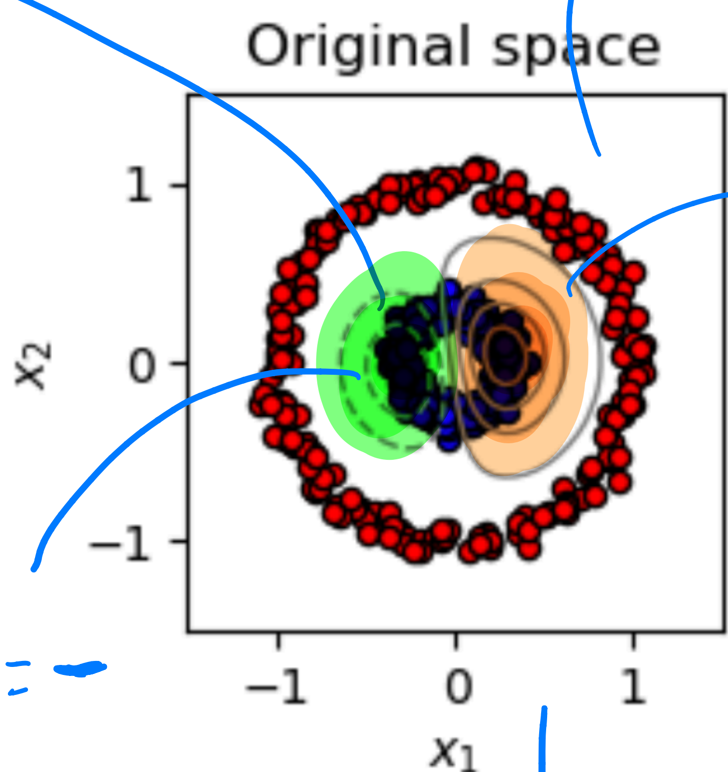
$M = \infty$

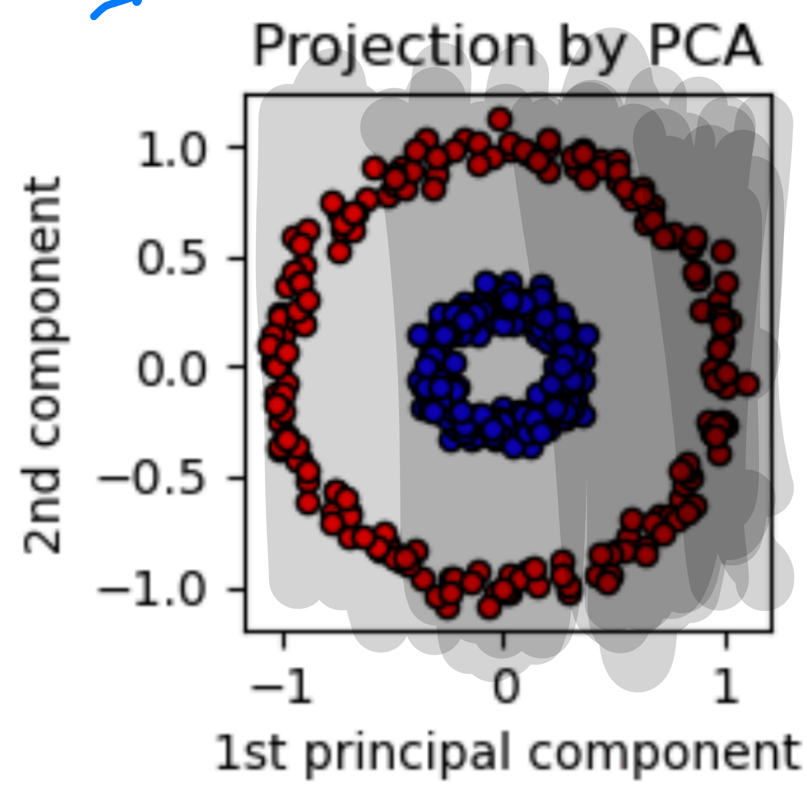
- Kernel trick: use some defined kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ without explicitly defining the basis functions $\boldsymbol{\phi}$

Powerful! : implicitly work with infinite dimensional features

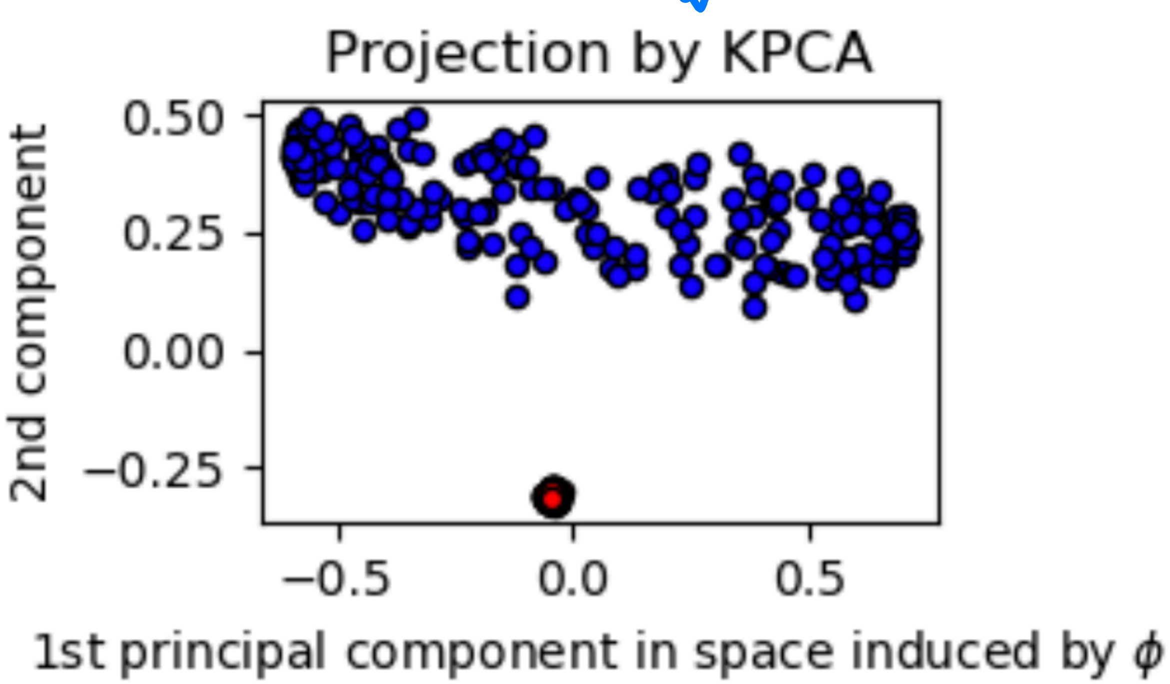
Example

$z_1(\underline{x})$ $z_1 = 0$ $z_1 = +$ $z_1 = -$





Z_2

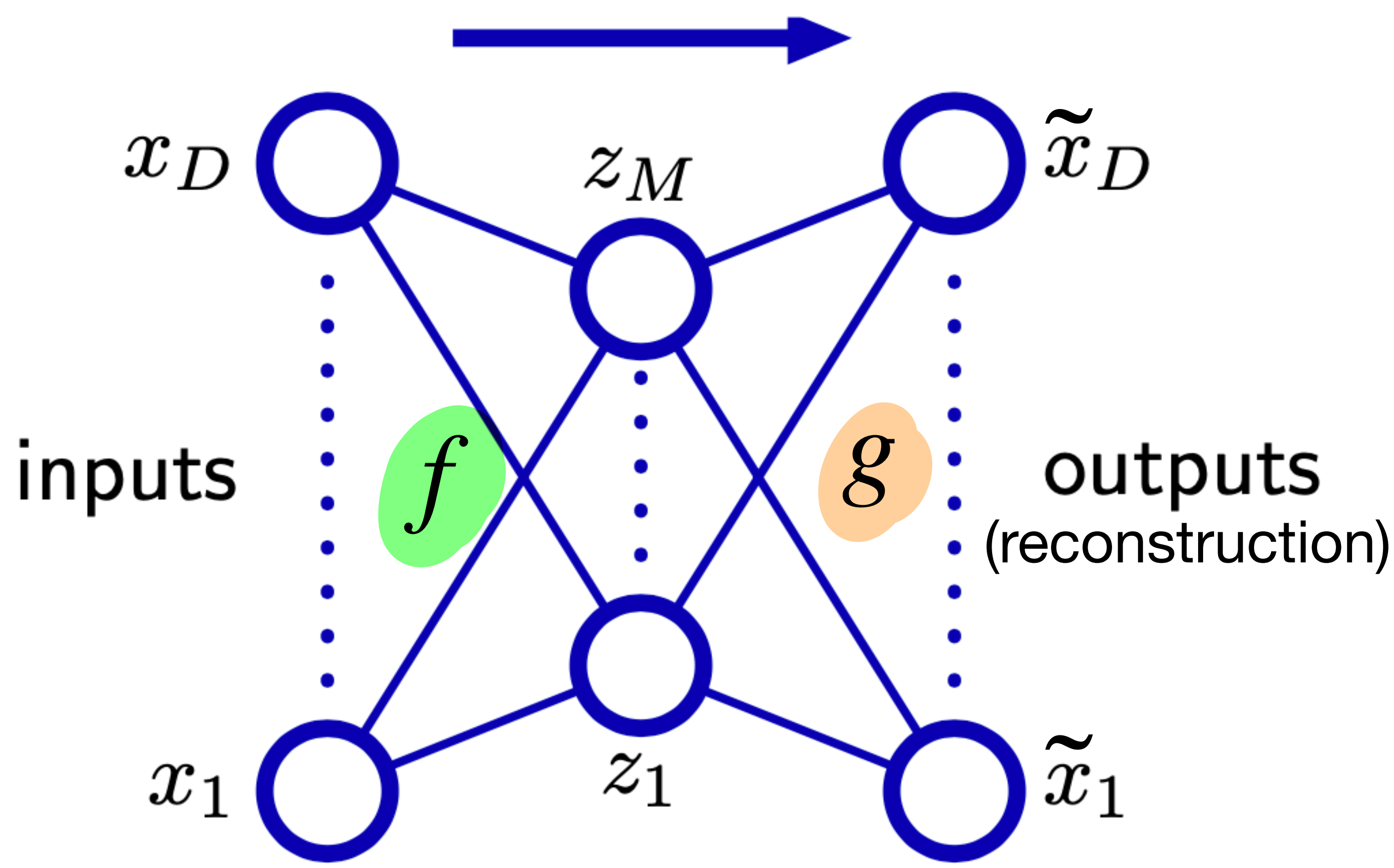


z_1

- With RBF
- $k(\mathbf{x_i}, \mathbf{x_j}) = e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}$
- Source:
  Kernel-PCA from
  http://scikit-learn.org

Auto-encoders (auto-associative neural nets)

- Non-linear dimensionality reduction with **neural networks**
- Maps:
  - Encoder (projection): $\mathbf{z} = f(\mathbf{x})$
  - Decoder (reconstruction): $\tilde{\mathbf{x}} = g(\mathbf{z})$
  - Both are neural networks
- Goal: minimize the reconstruction error



(Bishop 12.18)

Autoencoder objective

- Minimize the error between original and reconstructed input:

$$\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - g_{\mathbf{w}'}(\underbrace{f_{\mathbf{w}}(\mathbf{x}_n)}_{z})\|^2$$

- Non-linear, no closed form solution, solve via SGD!

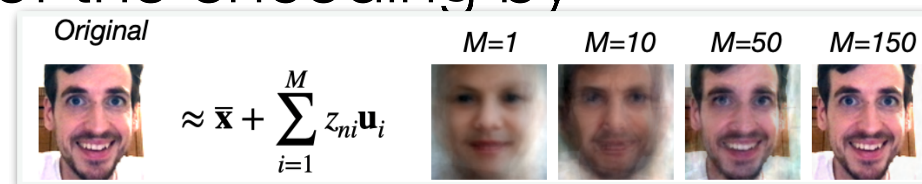
- Recall minimum error viewpoint of PCA:

$$f(\mathbf{x}) = \overset{W^{(1)}}{\mathbf{U}_{\mathbf{M}}^T}(\mathbf{x} - \overset{bias^{(1)}}{\bar{\mathbf{x}}}), \qquad g(\mathbf{x}) = \overset{W^{(2)}}{U_{\mathbf{M}}}\mathbf{z} + \overset{bias^{(2)}}{\bar{\mathbf{x}}}$$

- PCA <—> 2 layer autoencoder without activation functions

Autoencoder as generator

- As before, we could judge visually the quality of the encoding by looking at reconstructed images



- Bonus: we can use $\tilde{\mathbf{x}} = g(\mathbf{z})$ as a **generator of fake data** (images):
    - Train the autoencoder to get $\tilde{\mathbf{x}} = g(\mathbf{z})$
    - Sample a random $\mathbf{z}$ * $\sim p(z)$
    - Feed it into the generator to get fake image $\tilde{x} = g(\mathbf{z})$

**A probabilistic model needs to be defined before sampling, this definition needs to be part of the encoding/decoding optimization pipeline, see Variational Autoencoders (VAE) [Kingma & Welling, ICLR 2014]*

Autoencoder on MNIST

Which one is real?

… autoencoders in 2018

Trained on CelebA in [Patrini et al., Sinkhorn AutoEncoders, 2018]