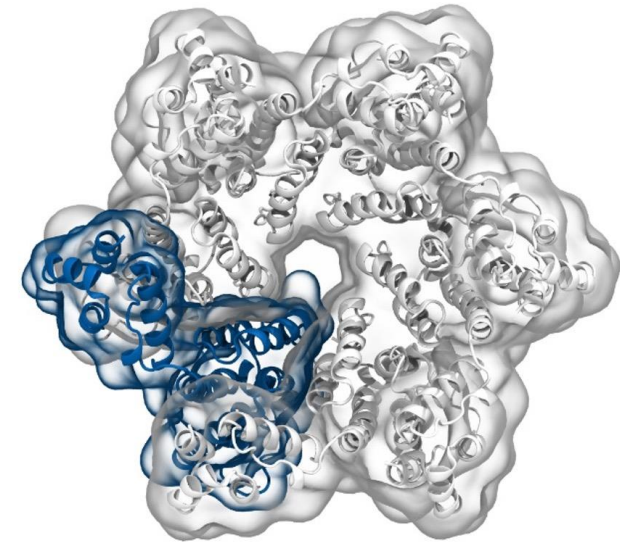
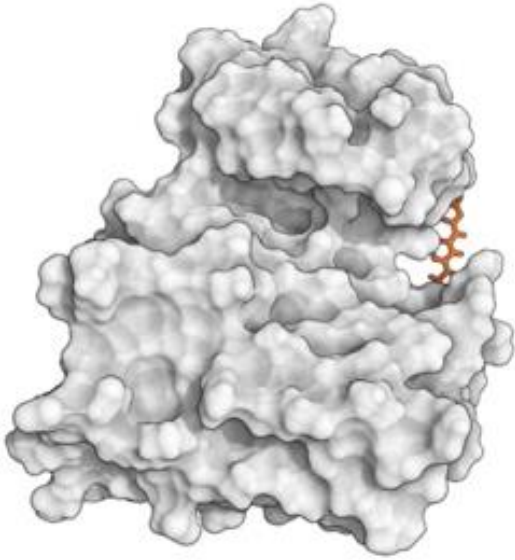


# Simulation of Biomolecules

## Dimensionality Reduction



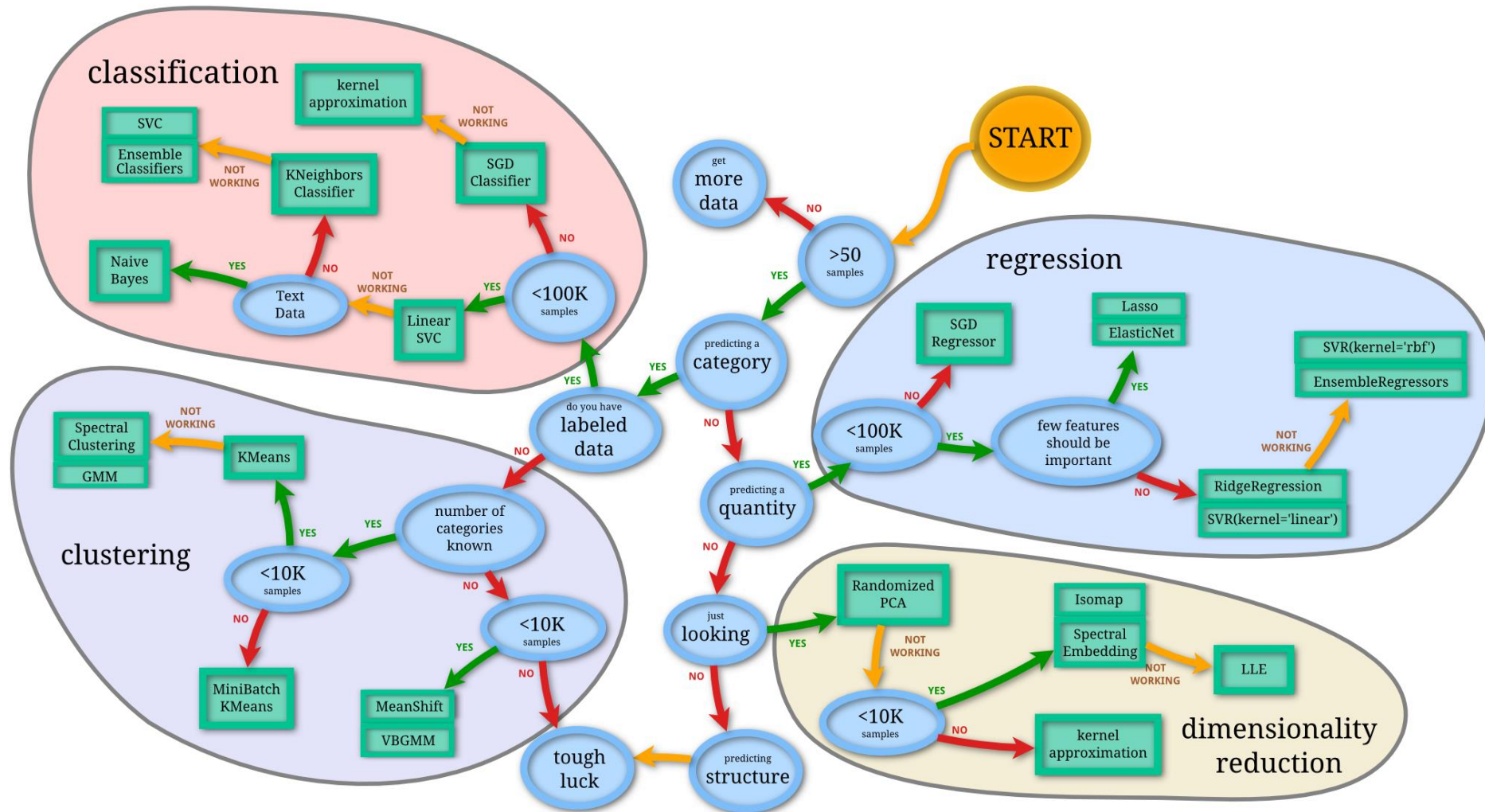
Dr Matteo Degiacomi  
Durham University

[matteo.t.degiacomini@durham.ac.uk](mailto:matteo.t.degiacomini@durham.ac.uk)

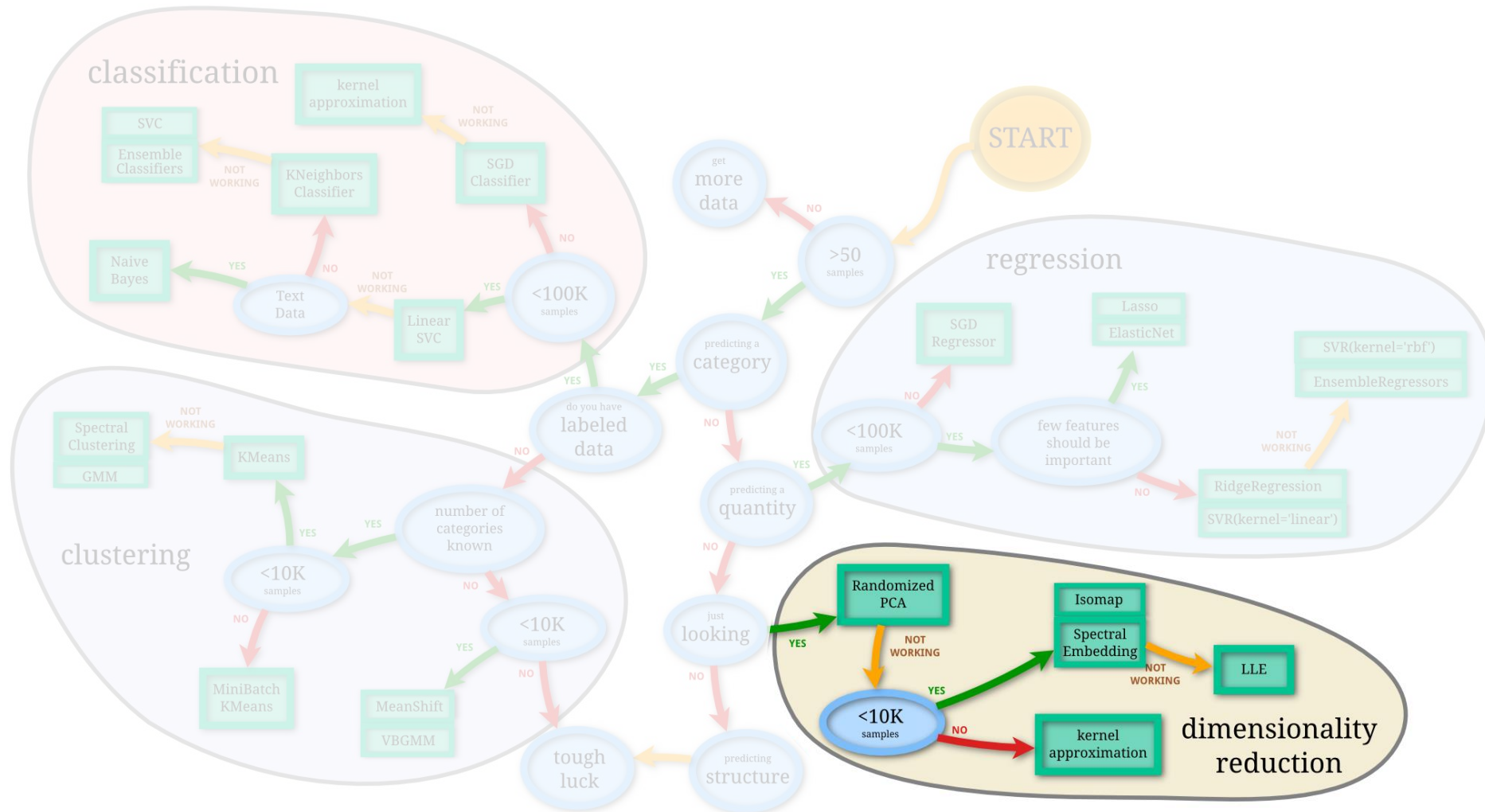
Dr Antonia Mey  
University of Edinburgh

[antonia.mey@ed.ac.uk](mailto:antonia.mey@ed.ac.uk)

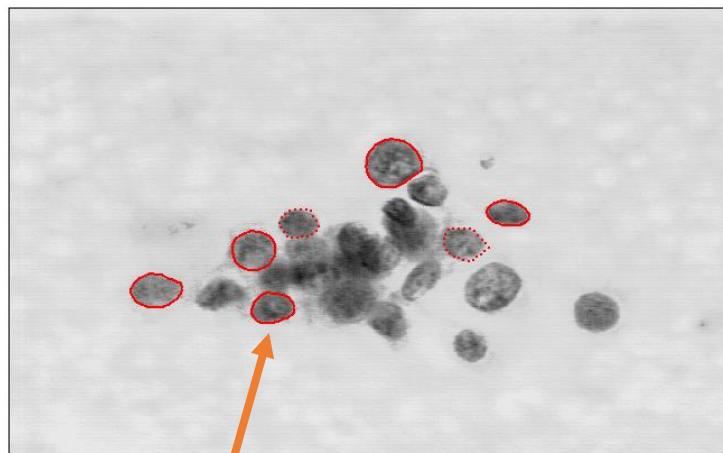
# The Data Mining world



# The Data Mining world

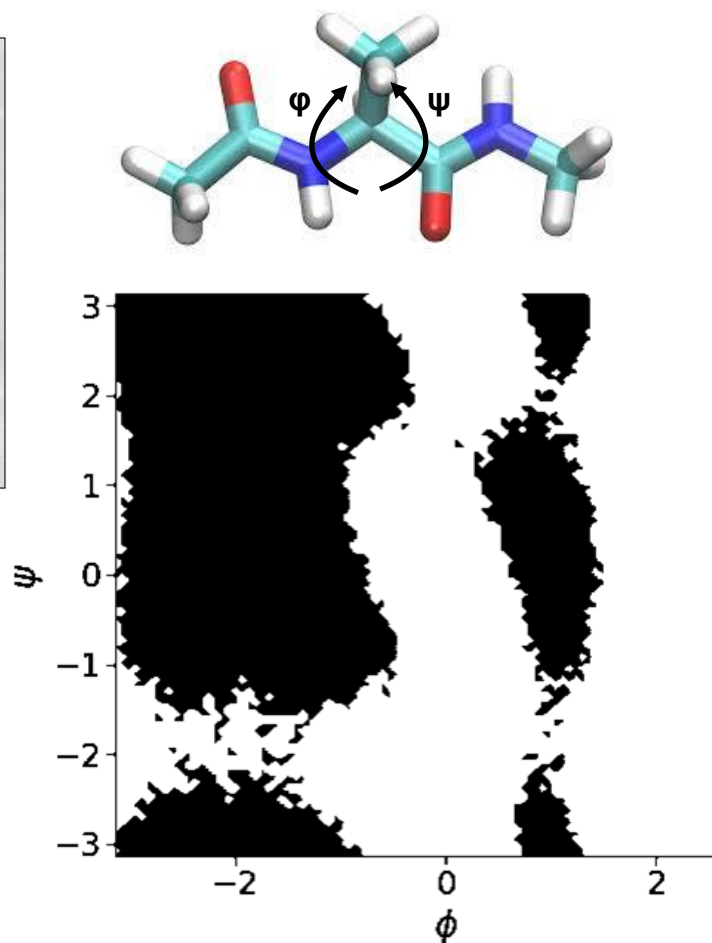


# *features* are possible ways to represent data

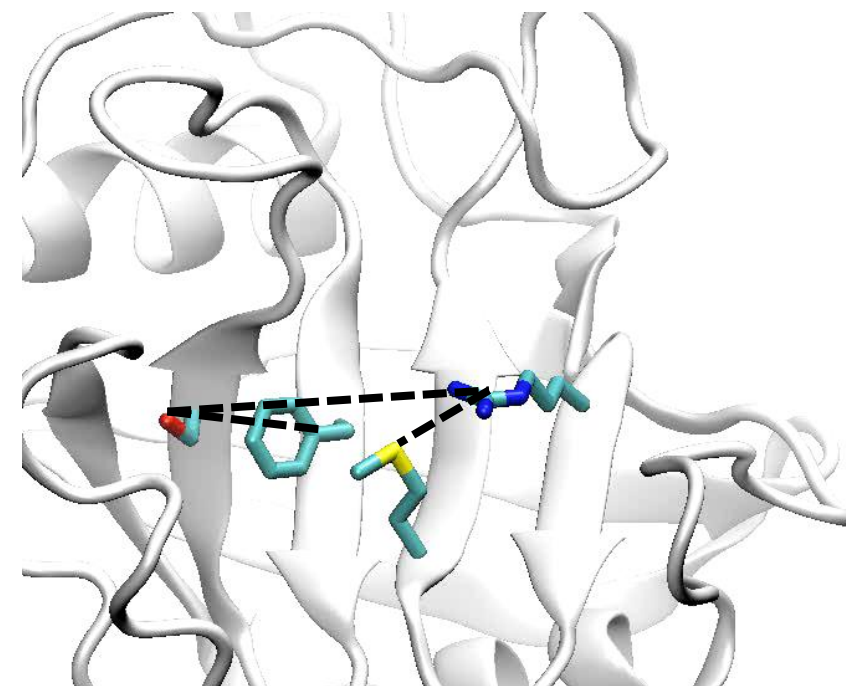


Sphericity (**1**)

Pixels colour  
(28x28 = **784**)



Torsional angles (**2**)



atomic positions (**459**)  
atomic distances (**3**)  
4



# Not all features are useful

**Task: predict the weather in Edinburgh  
using historical data**

data = {X, Y, Z} → sun, rain, snow



{Temperature (C),  
~~Temperature (K)~~,  
Humidity (g/m<sup>3</sup>)}

2 decorrelated  
features

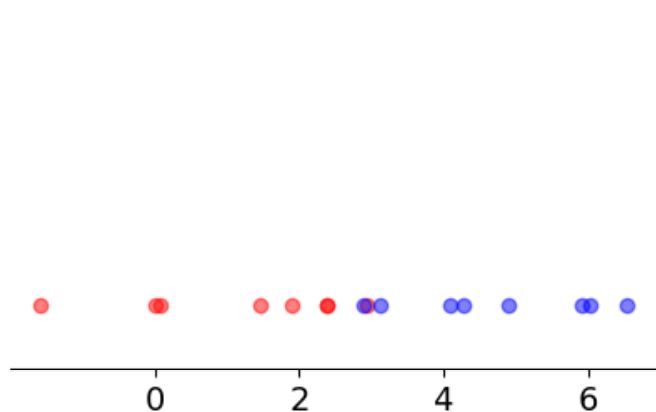
{Temperature (C),  
~~Swiss cheese export (£)~~,  
Humidity (g/m<sup>3</sup>)}

2 relevant  
features

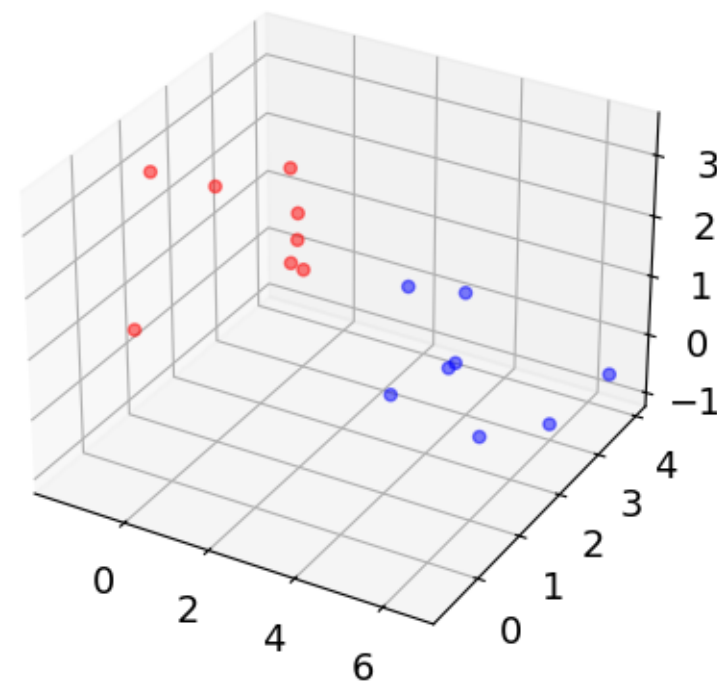
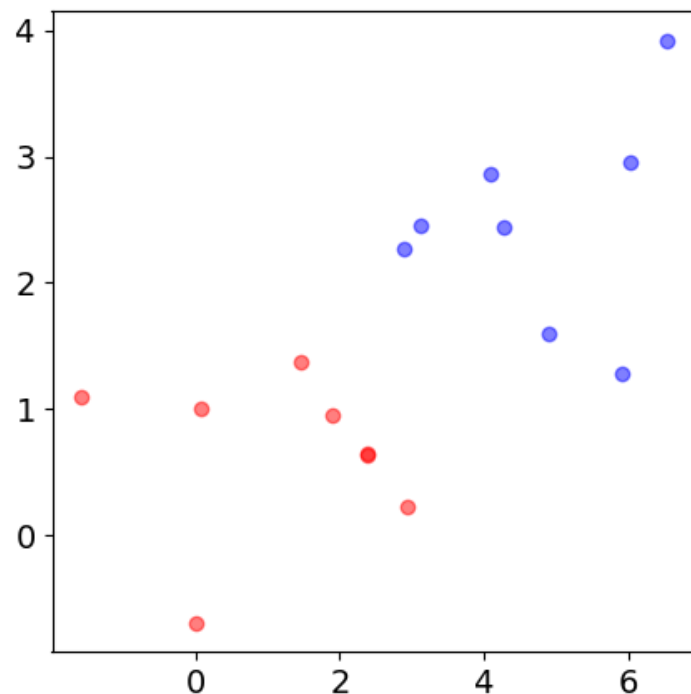
{Avg. gas expenditure (£),  
Heat strokes (#),  
Slipping accidents (#),  
Sunscreen sold (£)}

4 features connected to  
another quantity:  
temperature

# Curse of dimensionality



$$\|x\|_2 = \sqrt{\sum_{i=1}^N x_i^2}$$

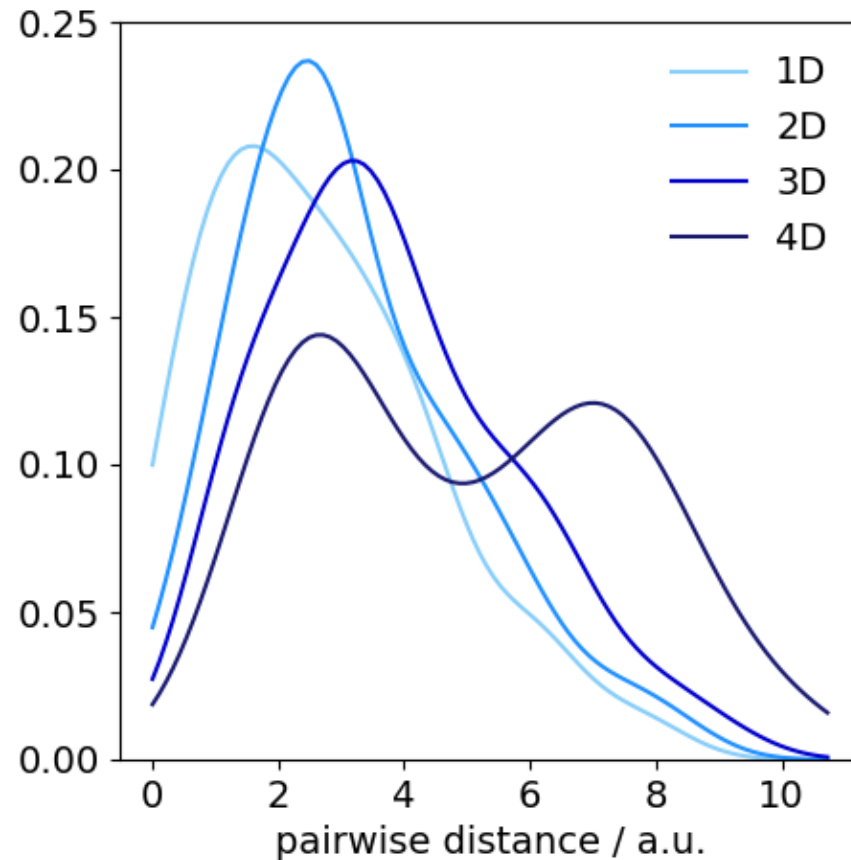


Distances between  $M$  data points  $x \in \mathbb{R}^N$  increase, when  $N$  increases

**Problem:** less data density increases uncertainty on underlying data structure

# [Extra] Curse of dimensionality

$$\|x\|_2 = \sqrt{\sum_{i=1}^N x_i^2}$$



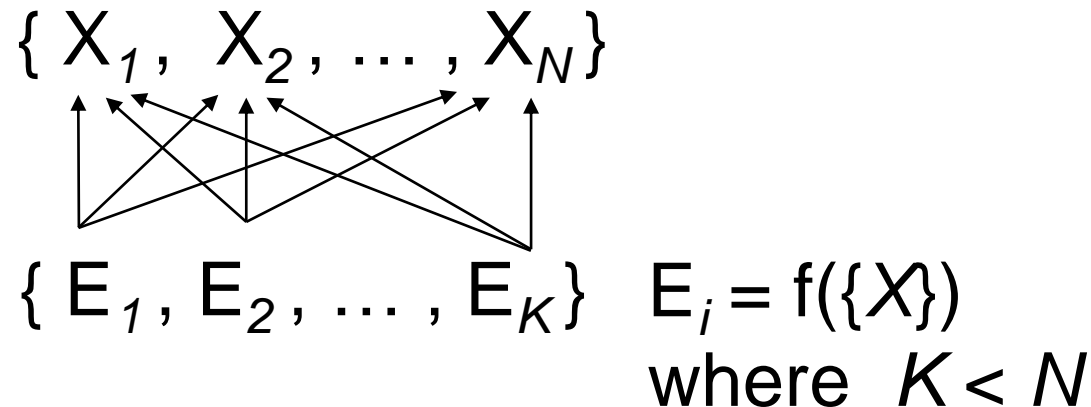
Distribution of pairwise distances between points shown in previous slide

Distances between  $M$  data points  $x \in \mathbb{R}^N$  increase, when  $N$  increases

**Problem:** less data density increases uncertainty on underlying data structure

# Reducing features increases data density

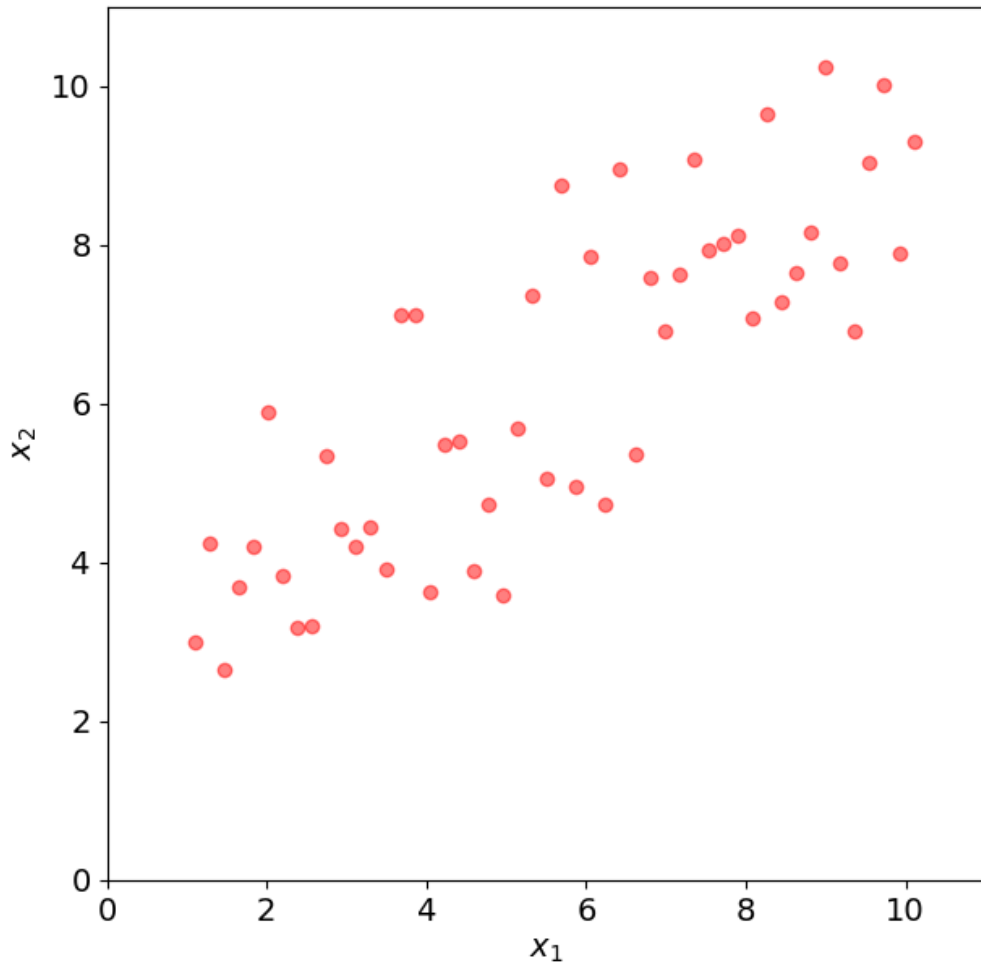
- Chose appropriate features [expert user]
- Remove features
- Find a lower dimensional representation  $E$  of features  $X$





# Principal Components Analysis (PCA)

Let  $\mathbf{X}$  a dataset of  $M$  datapoints in  $N$  dimensions (here,  $M=50$  and  $N=2$ )



- Center data:

$$\mathbf{X}' = \mathbf{X} - \mu$$

- Compute data covariance matrix  $\mathbf{C}$ :

$$c_{i,j} = \frac{1}{M} \sum_{k=1}^M \mathbf{x}'_i \mathbf{x}'_j$$

- Calculate eigenvalue decomposition:

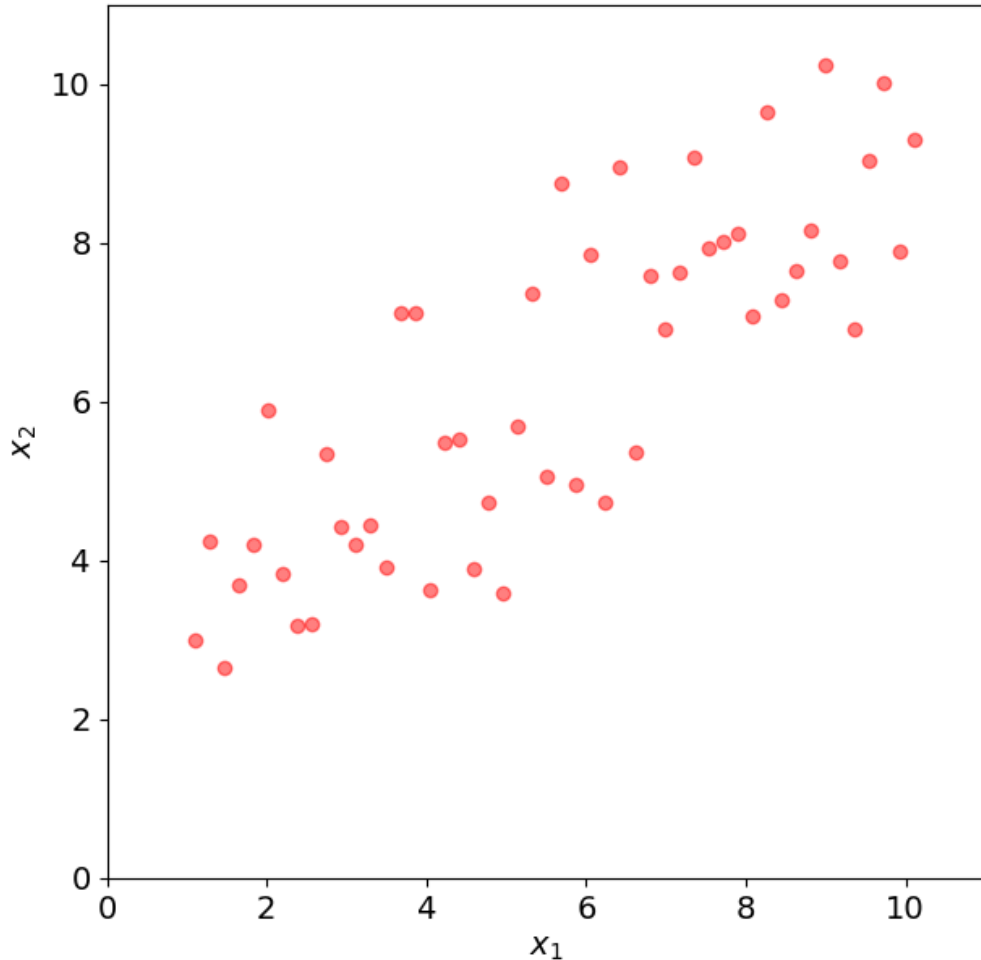
$$\mathbf{C} = \mathbf{V} \boldsymbol{\lambda} \mathbf{V}^{-1}$$

$N \times N$  matrix of  
**eigenvectors**

$N \times N$  diagonal matrix of  
**eigenvalues**

# [Extra] Principal Components Analysis (PCA)

Let  $\mathbf{X}$  a dataset of  $M$  datapoints in  $N$  dimensions (here,  $M=50$  and  $N=2$ )



An eigenvector  $\mathbf{v}$  of  $\mathbf{C}$  respects:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}$$

Find eigenvalues as the roots of the characteristic polynomial:

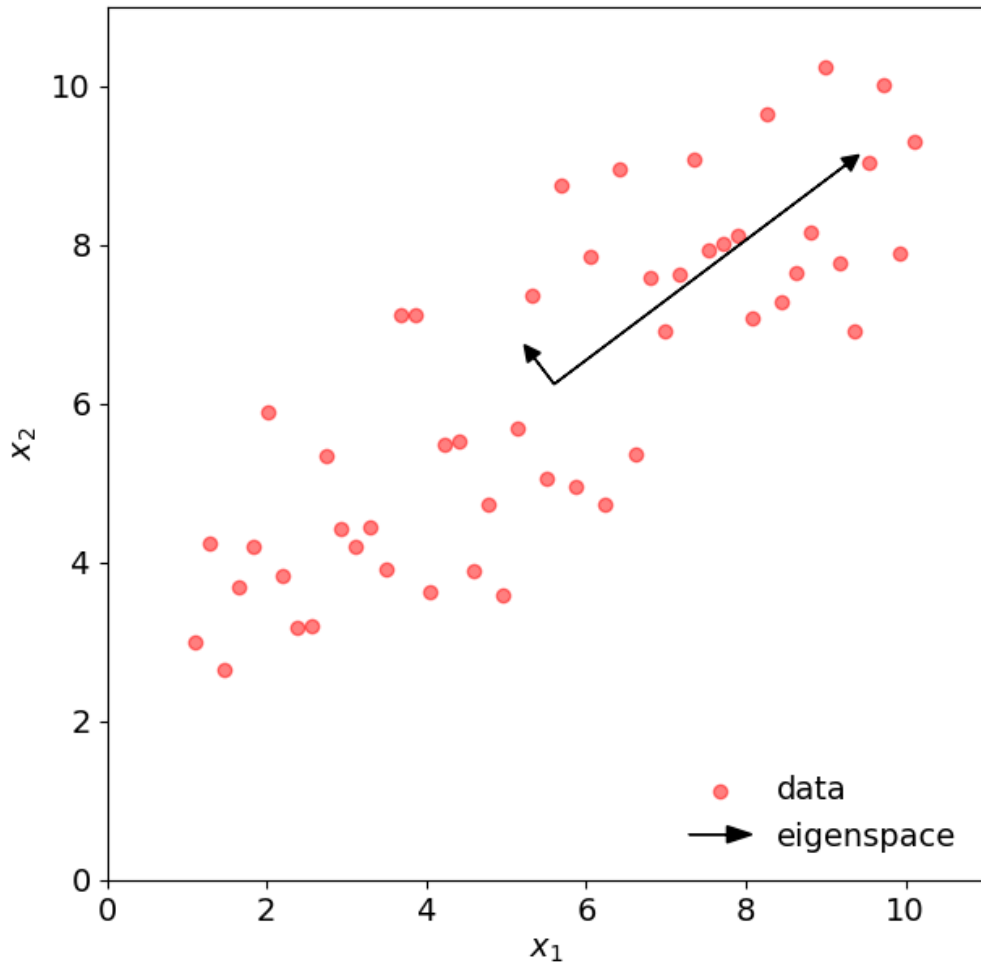
$$p(\lambda) = \det(\mathbf{C} - \lambda\mathbf{I}) = 0$$

The  $i$ -th eigenvector  $\mathbf{v}_i$  is found by solving:

$$\mathbf{C}\mathbf{v}_i = \lambda_i\mathbf{v}_i$$

# Principal Components Analysis (PCA)

Let  $\mathbf{X}$  a dataset of  $M$  datapoints in  $N$  dimensions (here,  $M=50$  and  $N=2$ )



$$\mathbf{C} = \mathbf{V}\boldsymbol{\lambda}\mathbf{V}^{-1}$$

$\mathbf{V} = [\mathbf{v}_1 \dots \mathbf{v}_N]$  eigenvectors:  
orthonormal base

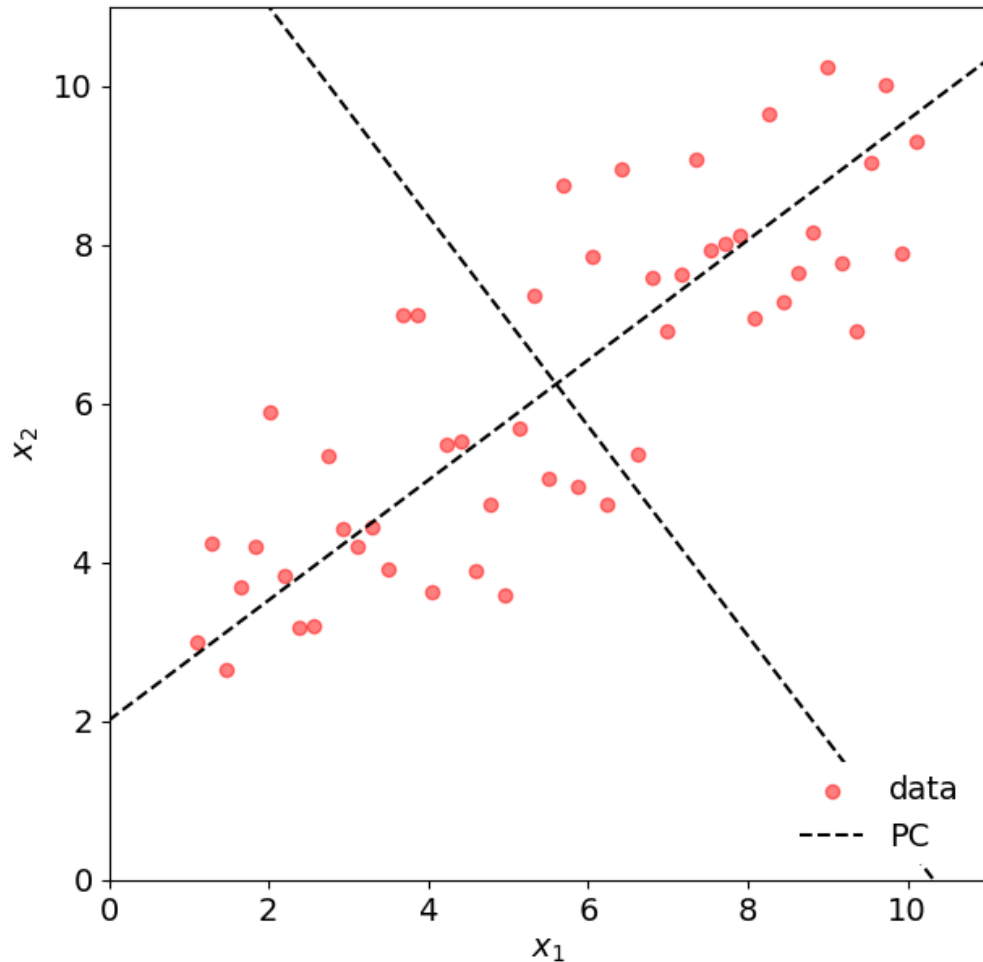
$\lambda$  eigenvalues: scalars defining the  
importance of each eigenvector  
Importance  $r_i$  of each eigenvector  $\mathbf{v}_i$  :

$$r_i = \frac{\lambda_i}{\sum \lambda}$$

*Sort  $\mathbf{V}$  and  $\lambda$  according to  $\lambda$*

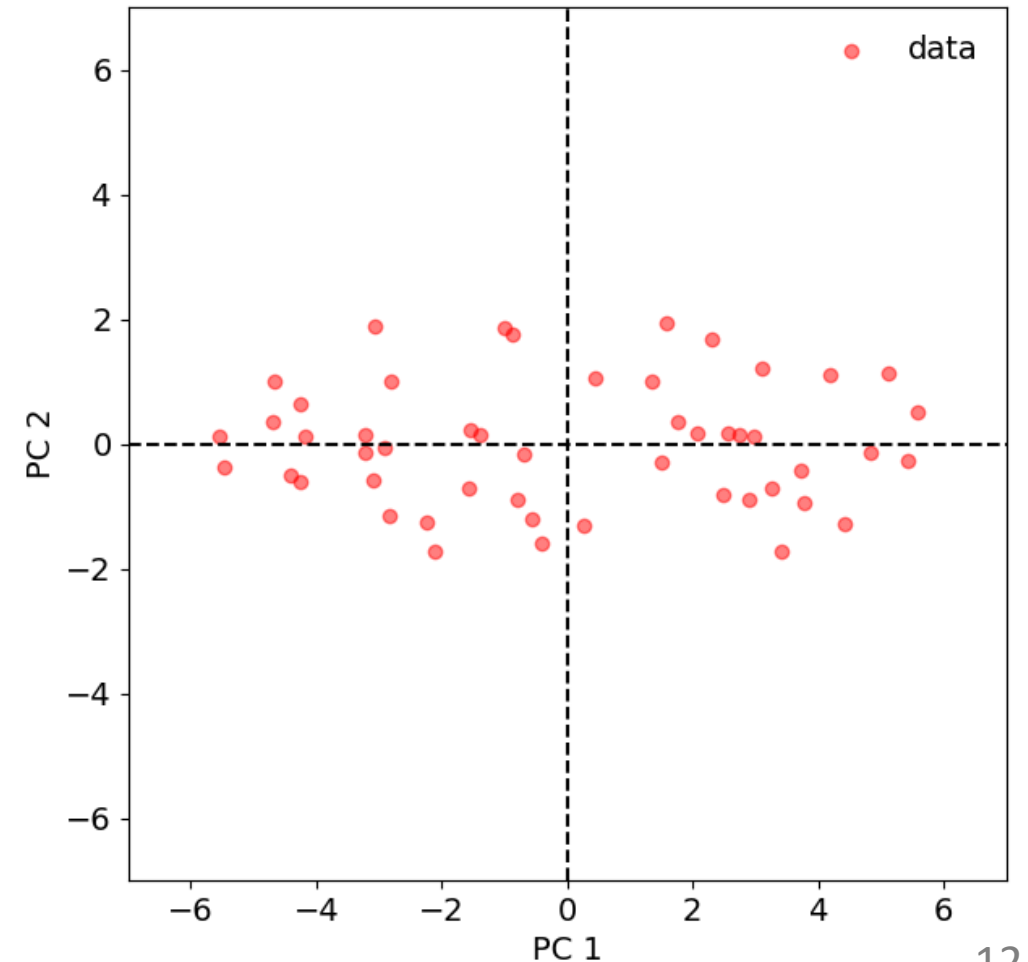
# Projection into the eigenspace

Let  $\mathbf{X}$  a dataset of  $M$  datapoints in  $N$  dimensions (here,  $M=50$  and  $N=2$ )



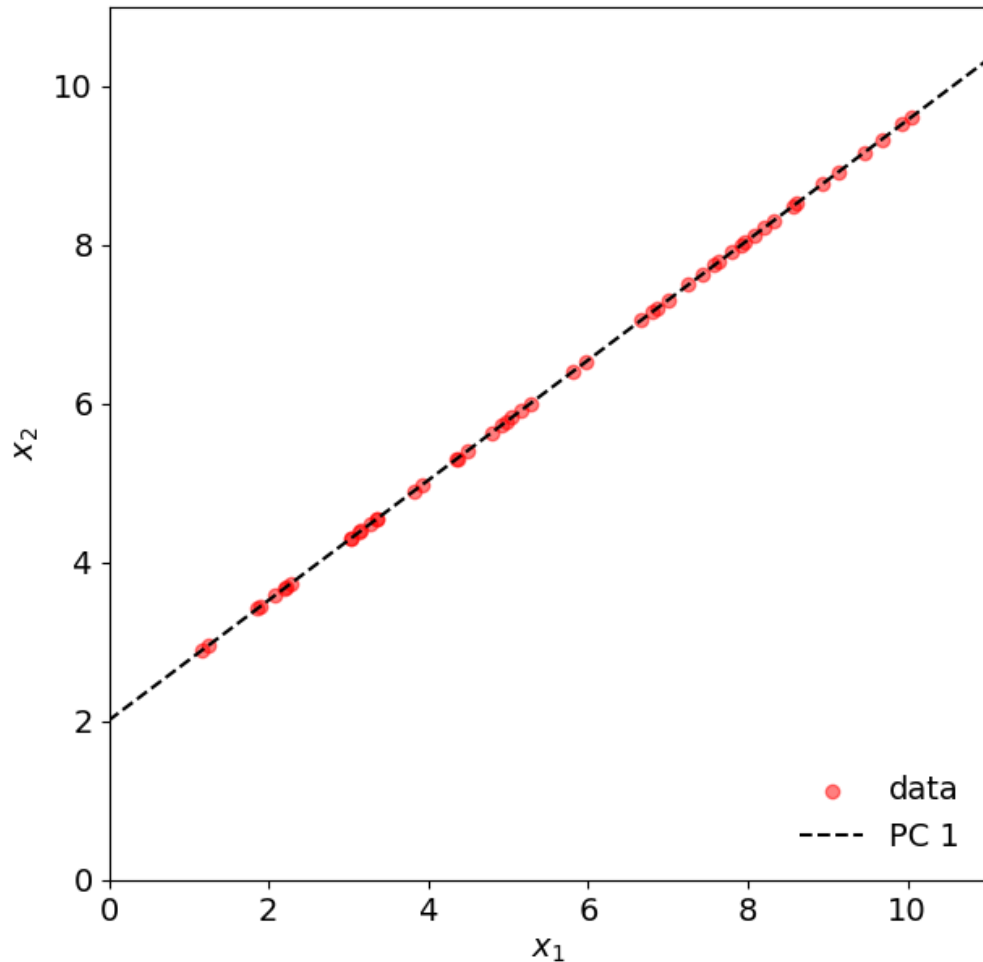
$$p = V^T(x - \mu)$$

transform



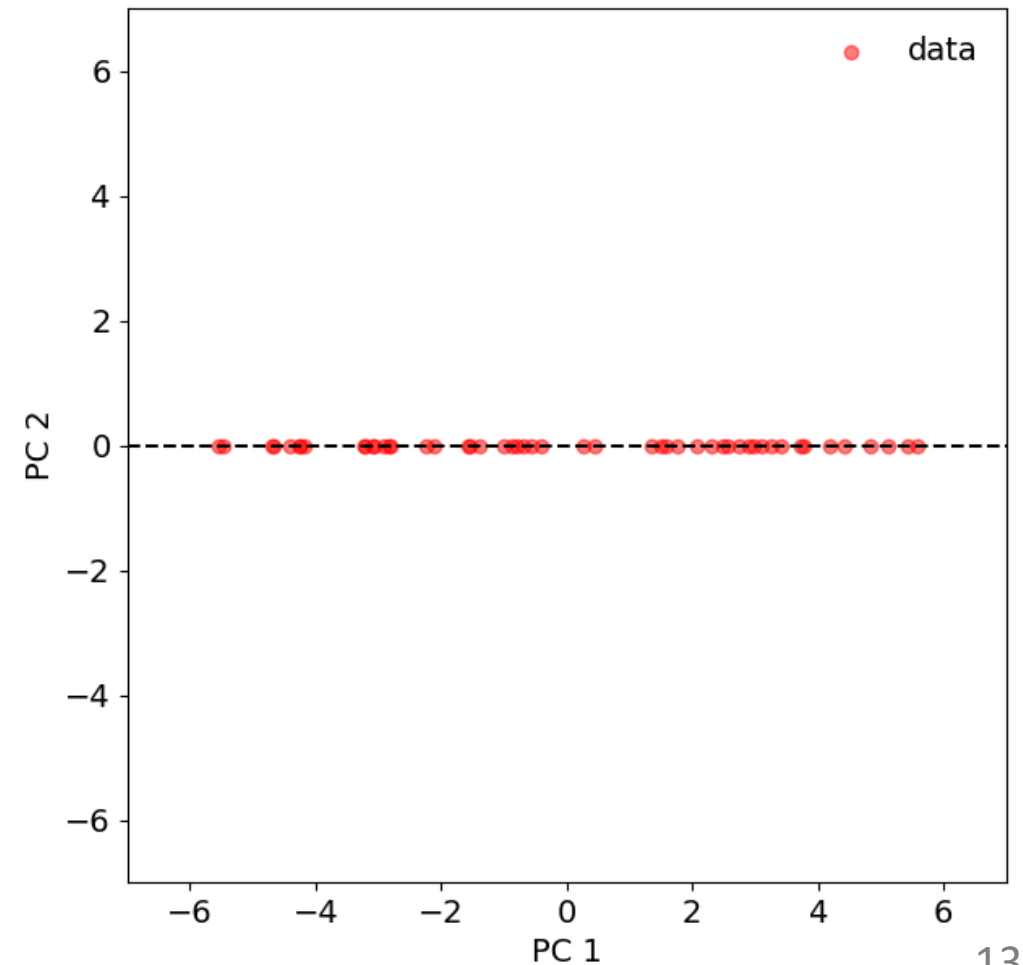
# Dimensionality reduction

Remove dimensions that least contribute to data variance



$$p = V^T(x - \mu)$$

transform

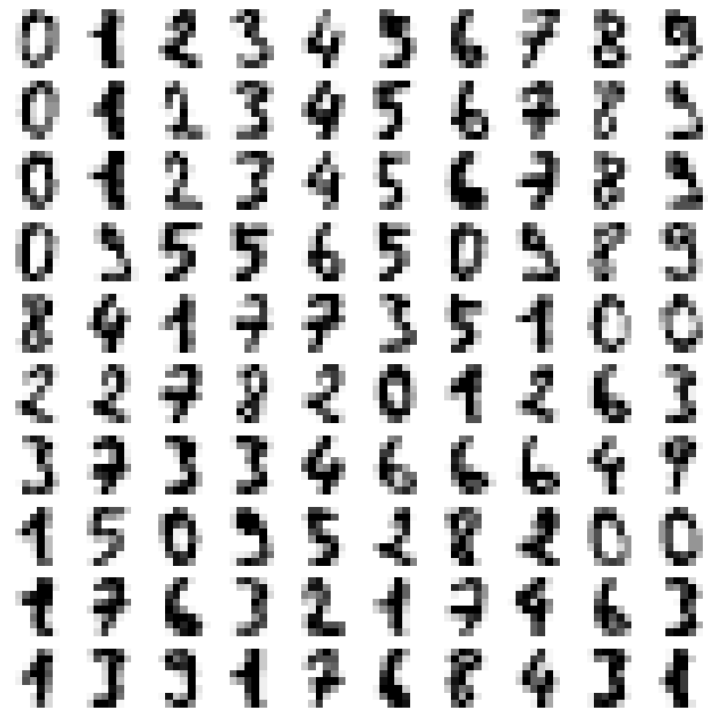




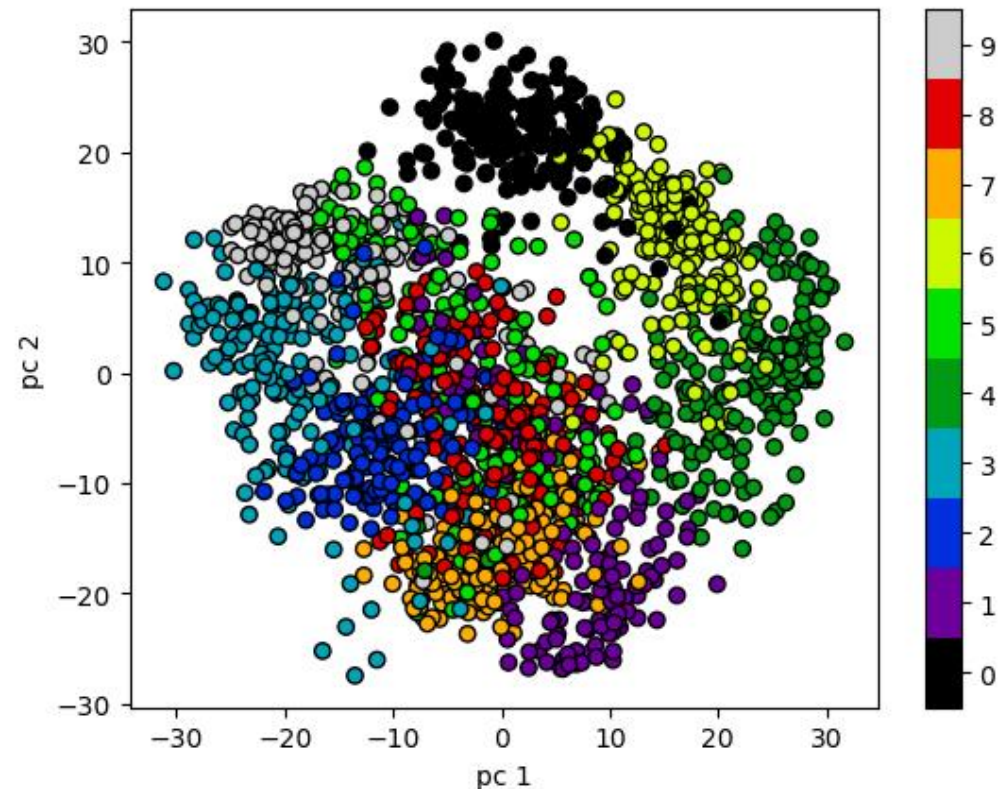
# [Example 1] Representing written digits

MNIST: database of written digits

Input: 8x8 pixel images  
(64-dimensional vector)

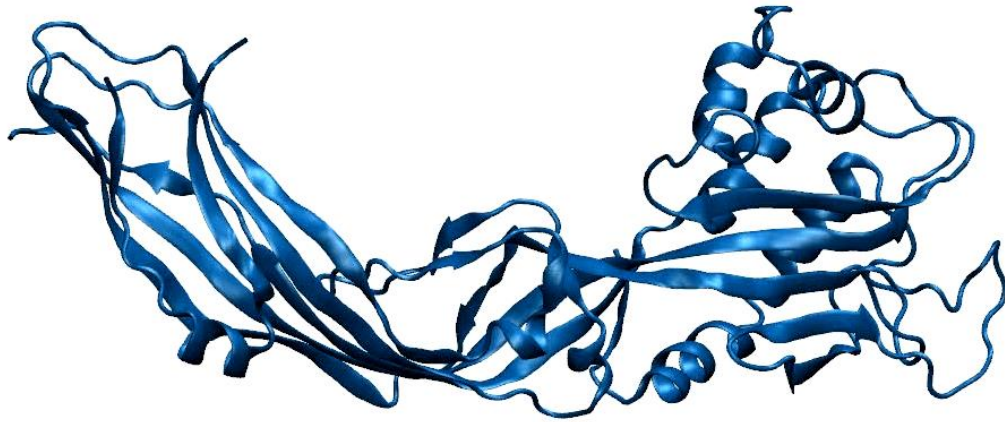


Projection in 2 principal components (PCs) separates some of the digits!

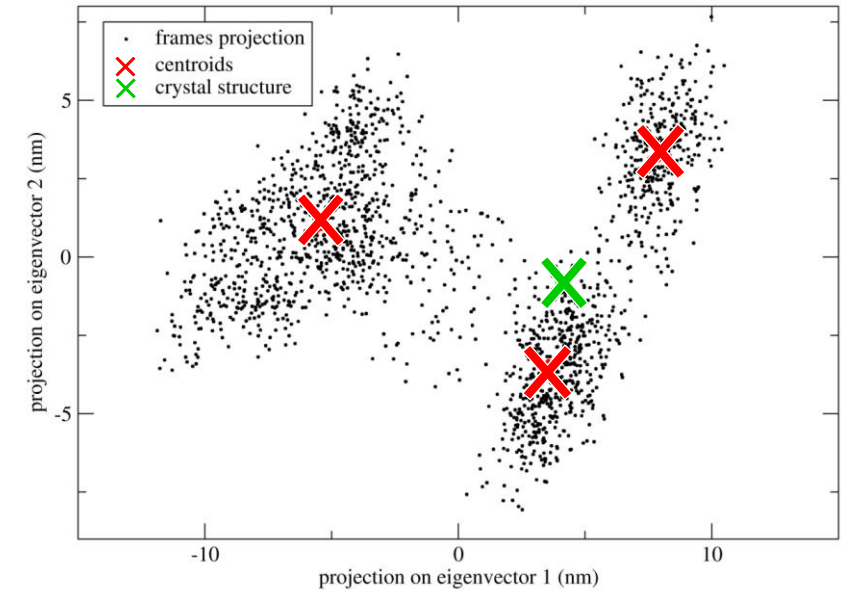


# [Example 2] Identifying dominant motions in proteins

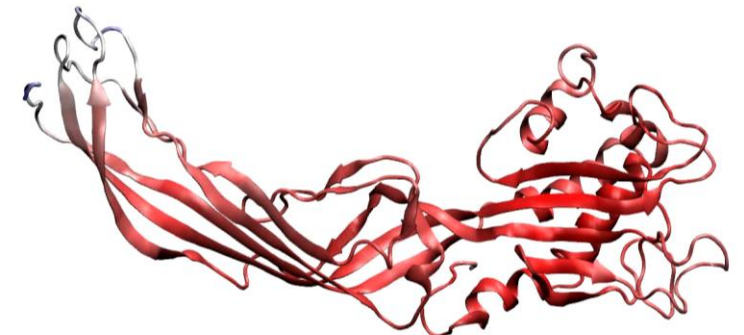
Protein MD simulation



Eigenspace of  $C_\alpha$  coordinates

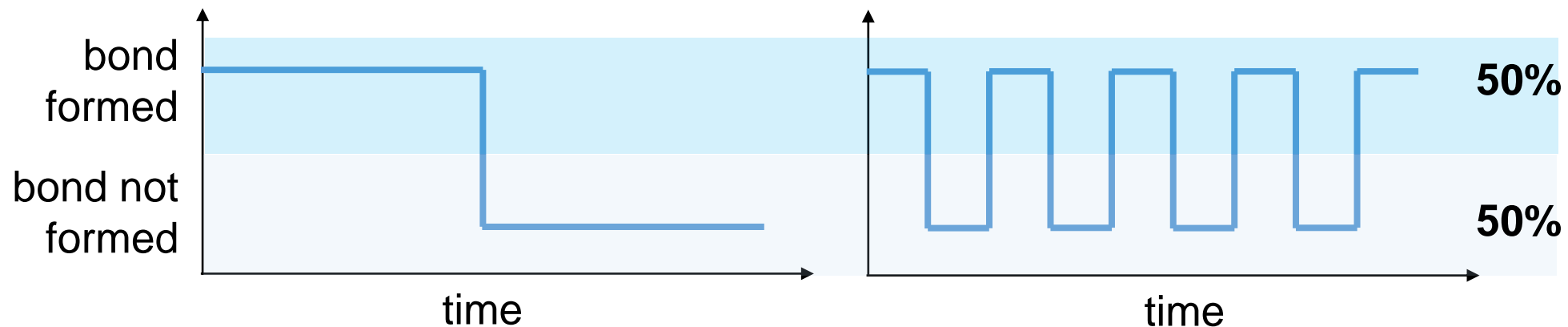


- Simulations are complex and noisy
- The first PCs capture large-scale collective motions, last ones capture noise



# largest-scale motion $\neq$ slowest motion

**Thought experiment reminder:** typically hydrogen bond is considered established if donor-acceptor distance  $< 2.5 \text{ \AA}$ , and donor-acceptor-hydrogen angle  $< 20^\circ$ .



- Biological function is often determined by the kinetics of a process (e.g., a specific conformational change).
- Two processes with different kinetics can have the same statistical distribution in terms of structure.
- PCA separates in terms of structure, not timescales!

# Time-lagged independent component analysis (tlICA)

tlICA is a **linear transform** similar to PCA

The transform is chosen such that, amongst all linear transforms, tlICA **maximizes the autocorrelation** of transformed coordinates.

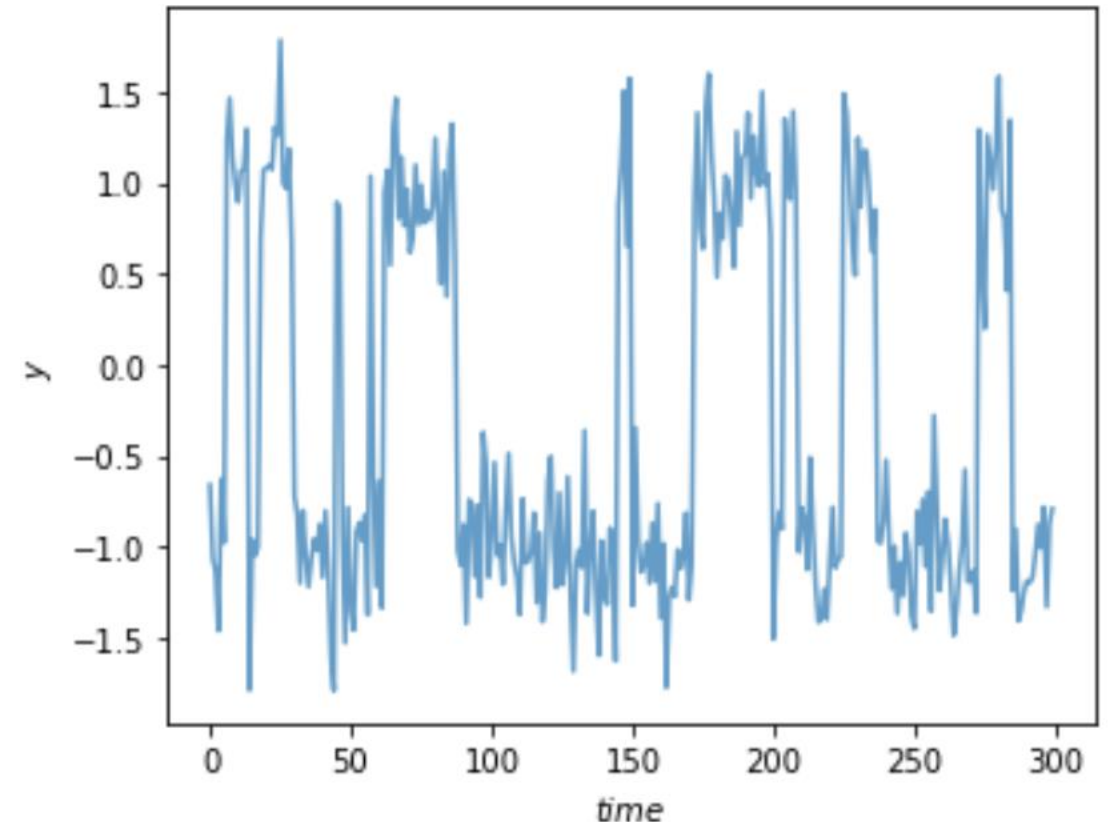
$$\mathbf{r}(t) = (r_i(t))_{i=1,\dots,D}$$

D-dimensional input data vector that is mean free, i.e.,  $\mathbf{r}(t) = \mathbf{r}(t) - \langle \mathbf{r}(t) \rangle_t$

Computing the covariance of the data at  $t = 0$  and  $t = \tau$  which is the lag-time chosen

$$c_{ij}(\tau) = \langle r_i(t)r_j(t + \tau) \rangle_t$$

enables computing two covariance matrices:  $\mathbf{C}(0)$  and  $\mathbf{C}(\tau)$



# Time-lagged independent component analysis (tlICA)

tlICA is a **linear transform** similar to PCA

The transform is chosen such that, amongst all linear transforms, tlICA **maximizes the autocorrelation** of transformed coordinates.

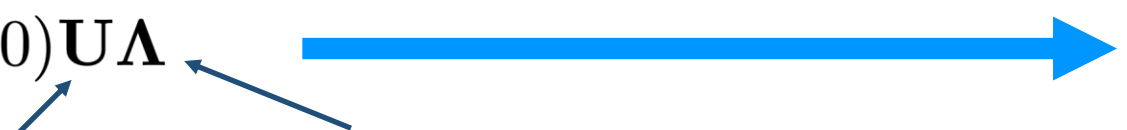
Entries of the covariance matrix can be computed as:

$$c_{ij}(\tau) = \frac{1}{N - \tau - 1} \sum_{t=1}^{N-\tau} r_i(t)r_j(t + \tau)$$

$\mathbf{C}(0)$  will be a symmetric matrix. The symmetry of  $\mathbf{C}(\tau)$  will need to be enforced with:

$$\mathbf{C}(\tau) = \frac{1}{2}(\mathbf{C}_d(\tau) + \mathbf{C}_d^T(\tau))$$

We can now solve the generalised eigenvalue problem:

$$\mathbf{C}(\tau)\mathbf{U} = \mathbf{C}(0)\mathbf{U}\mathbf{\Lambda}$$


Eigenvector matrix containing ICs

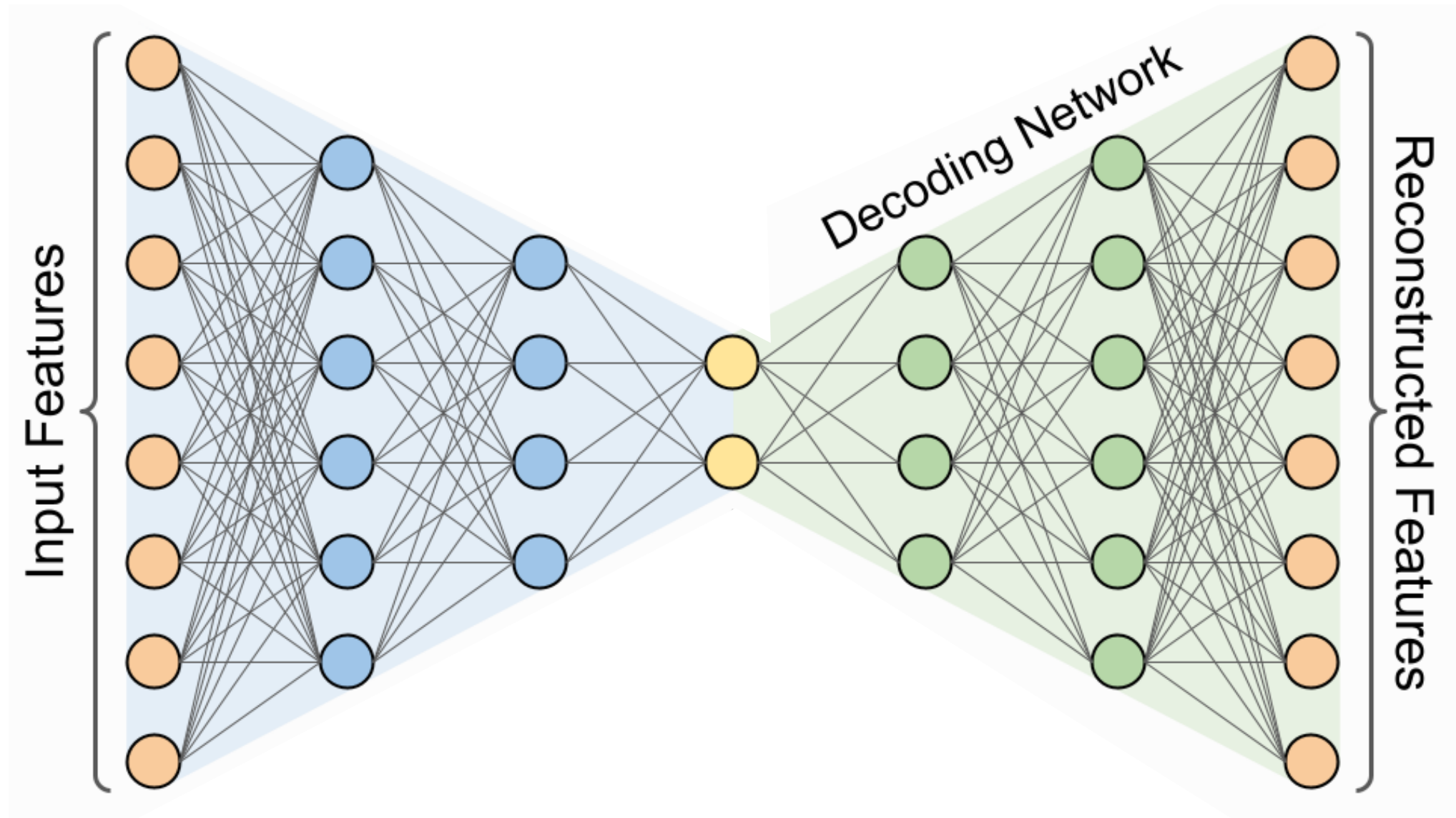
Diagonal matrix with eigenvalues

$$\mathbf{z}^T(t) = \mathbf{r}^T(t)\mathbf{U}$$

M columns of full rank  $\mathbf{U}$  for DR



# [Extra] Dimensionality reduction with neural networks



PCA dimensionality reduction is *interpretable*, that of t-SNE and neural networks is not.