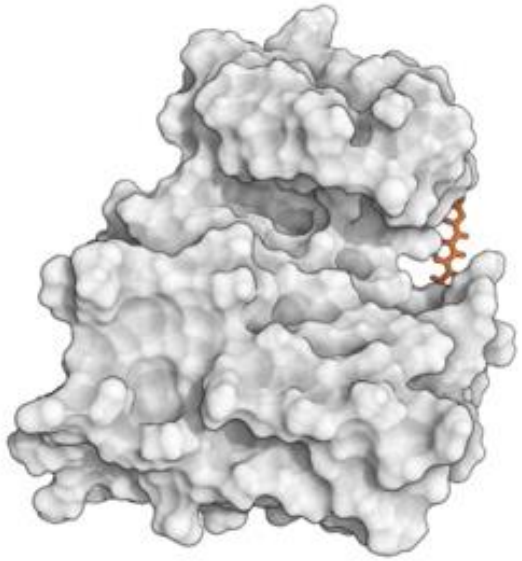
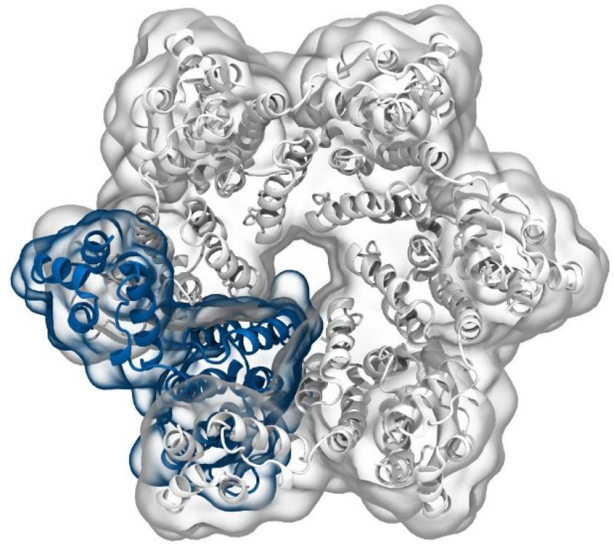
Simulation of Biomolecules



Simulation Analysis part 1

2024 CCP5 Summer School



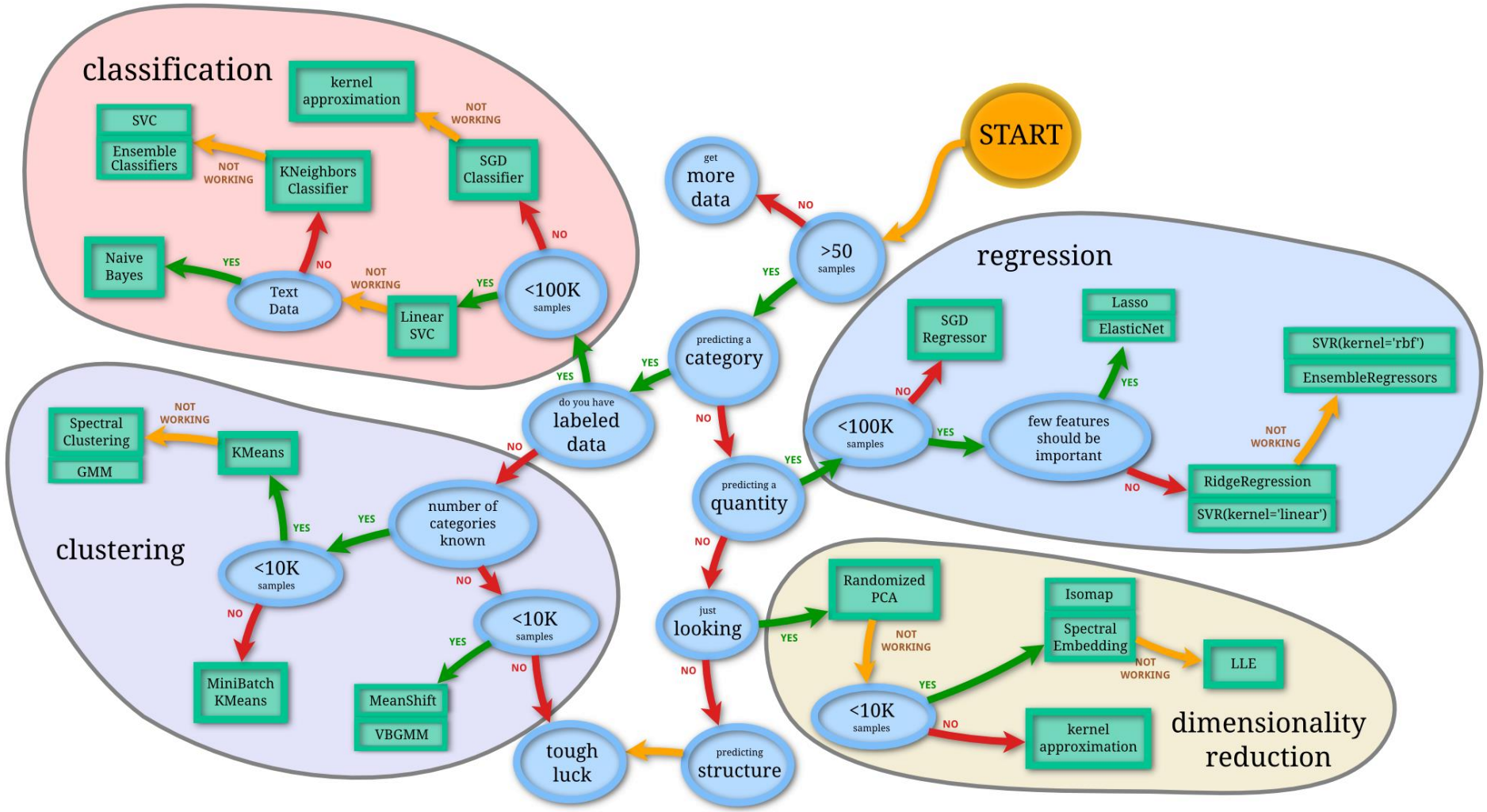
Dr Matteo Degiacomi
Durham University
matteo.t.degiacomi@durham.ac.uk

Dr Antonia Mey
University of Edinburgh
antonia.mey@ed.ac.uk

Large ecosystem of Python-based tools data analysis

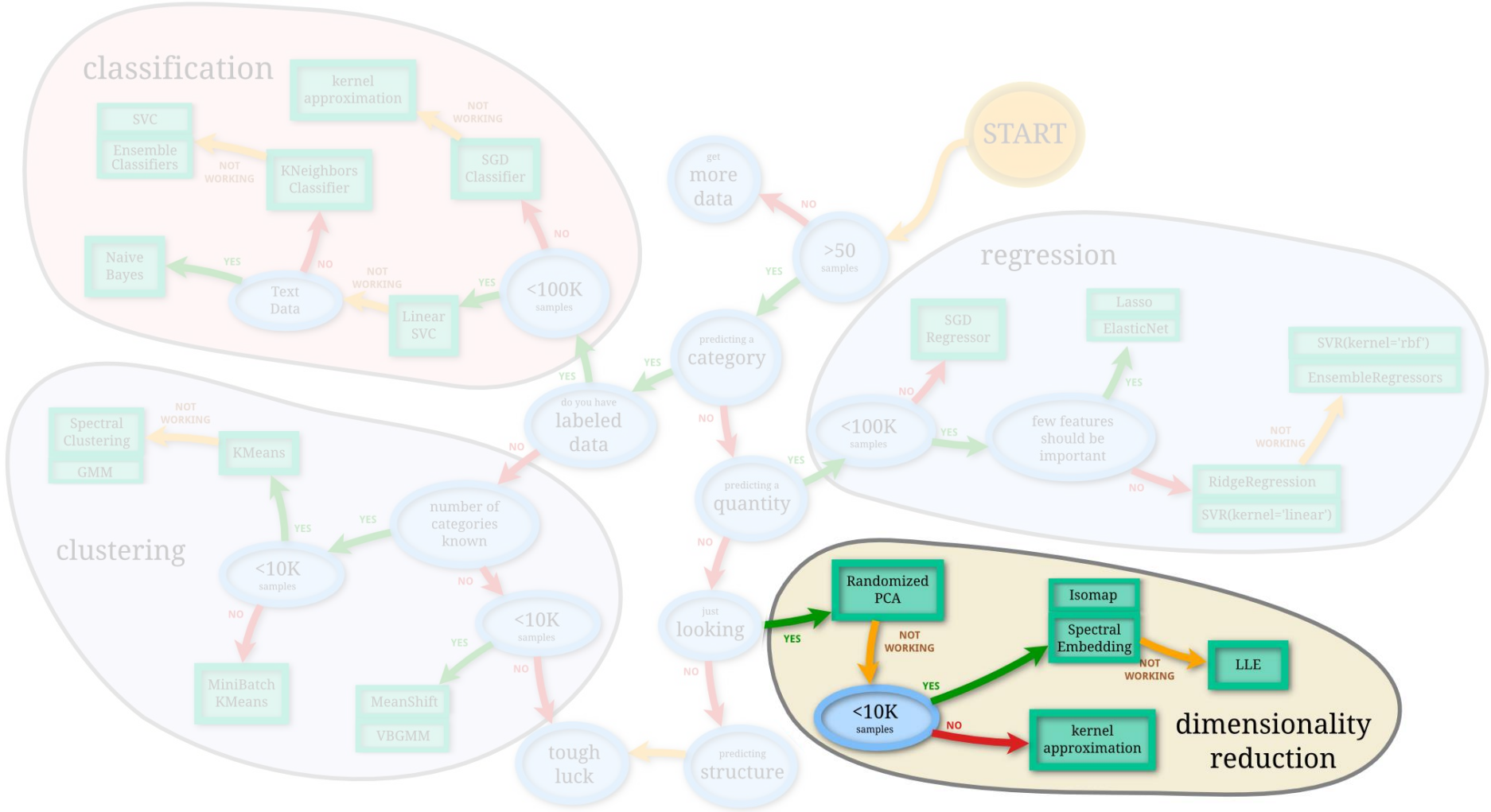
And many more…

The Data Mining world



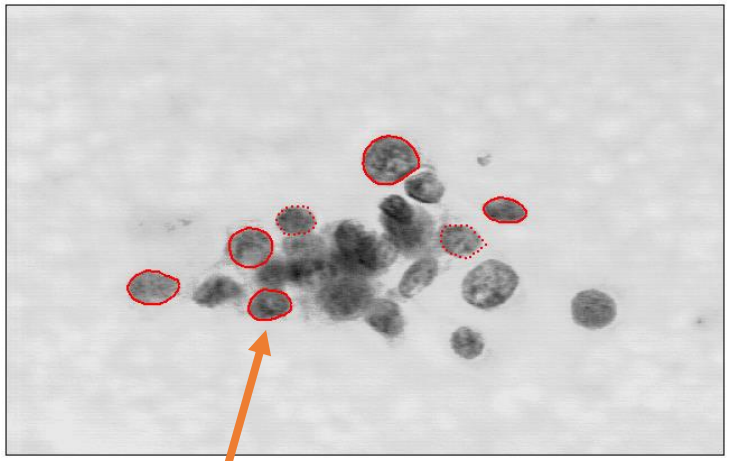
From **scikit-learn.org**

The Data Mining world



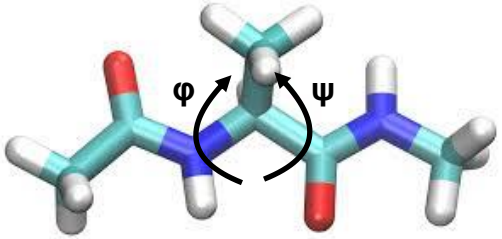
From **scikit-learn.org**

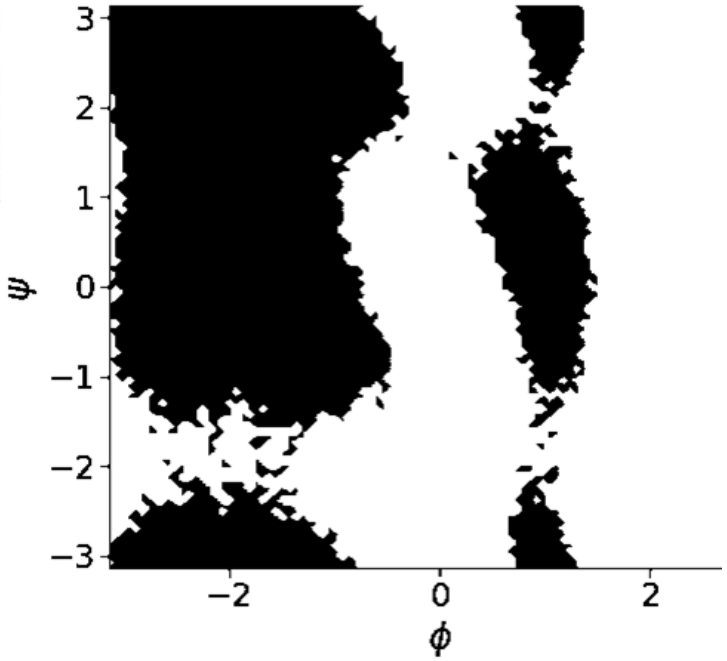
features are possible ways to represent data



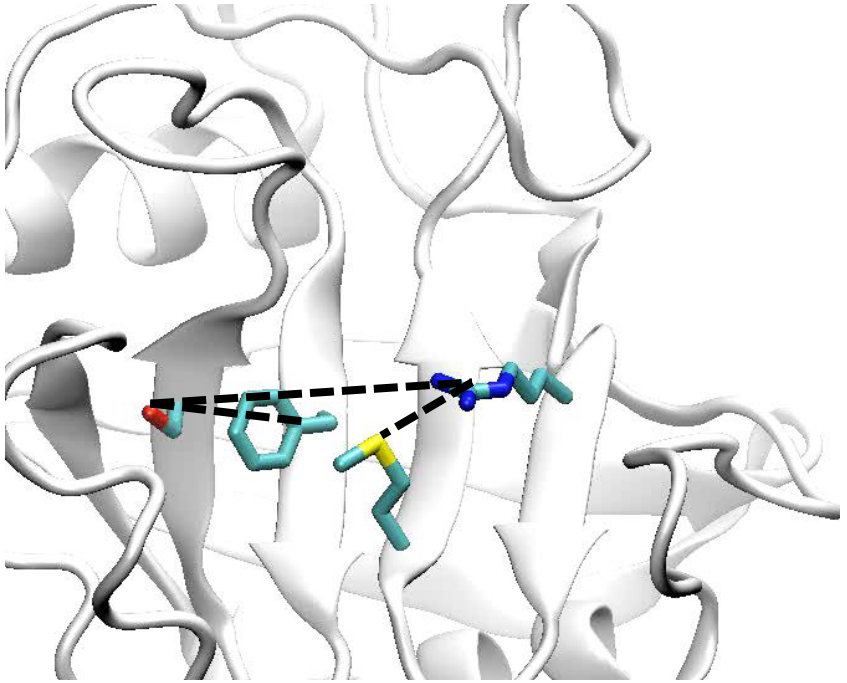
Sphericity (***1***)

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





Torsional angles (***2***)



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

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^3)}

2 decorrelated features

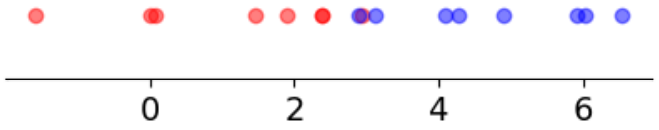
{Temperature (C),
~~Swiss cheese export (£),~~
Humidity (g/m^3)}

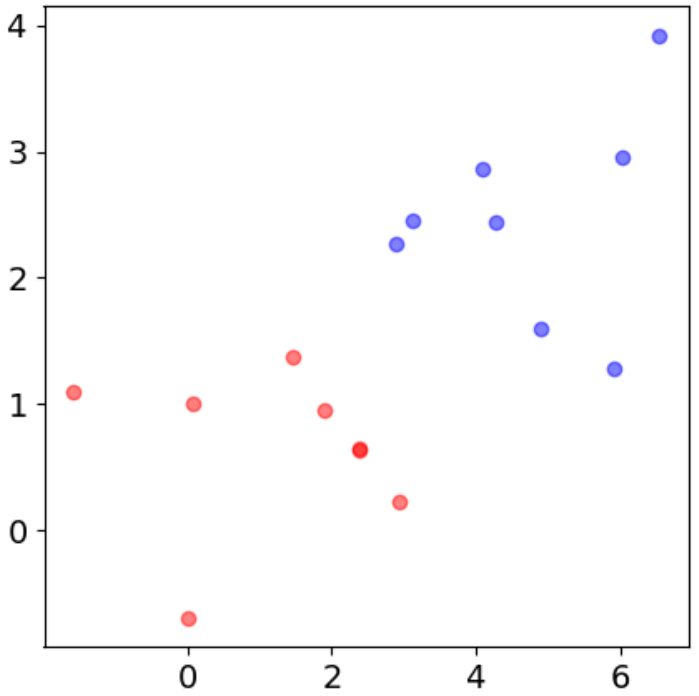
2 relevant features

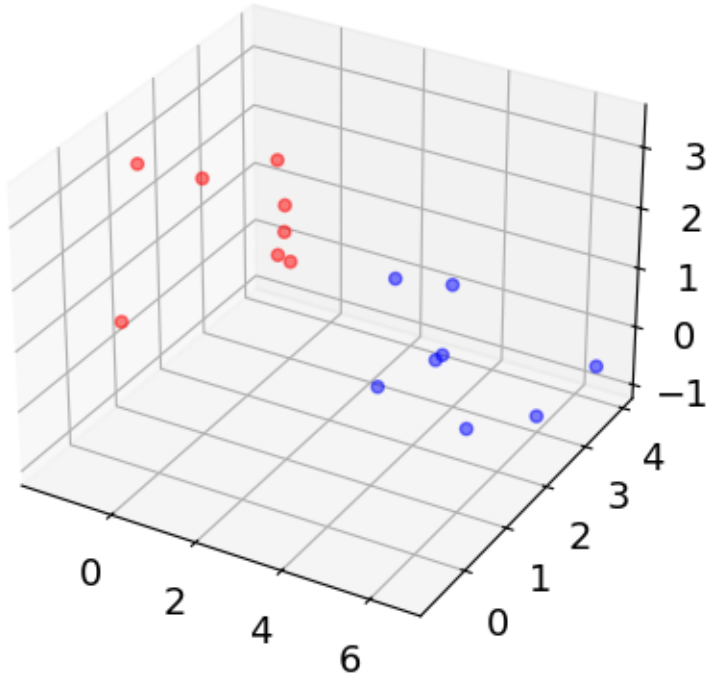
{Avg. gas expenditure (£),
Heat strokes (#),
Slipping accidents (#),
Sunscreen sold (£)}

4 features connected to another quantity temperature

Curse of dimensionality







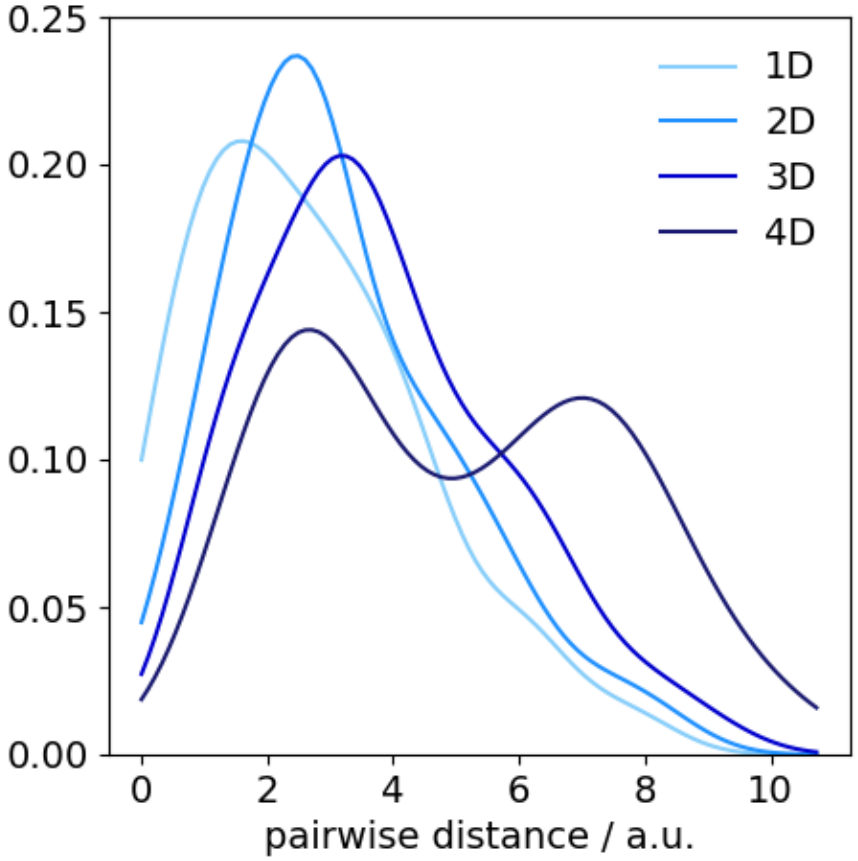
$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

Distances between M data points $\boldsymbol{x} \in \mathbb{R}^N$ 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

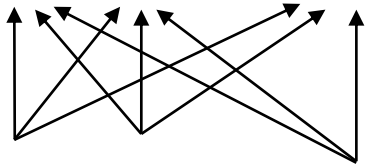
Distances between *M* data points $\boldsymbol{x} \in \mathbb{R}^N$ increase, when *N* increases

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*

$\{ X_1, X_2, \ldots, X_N \}$



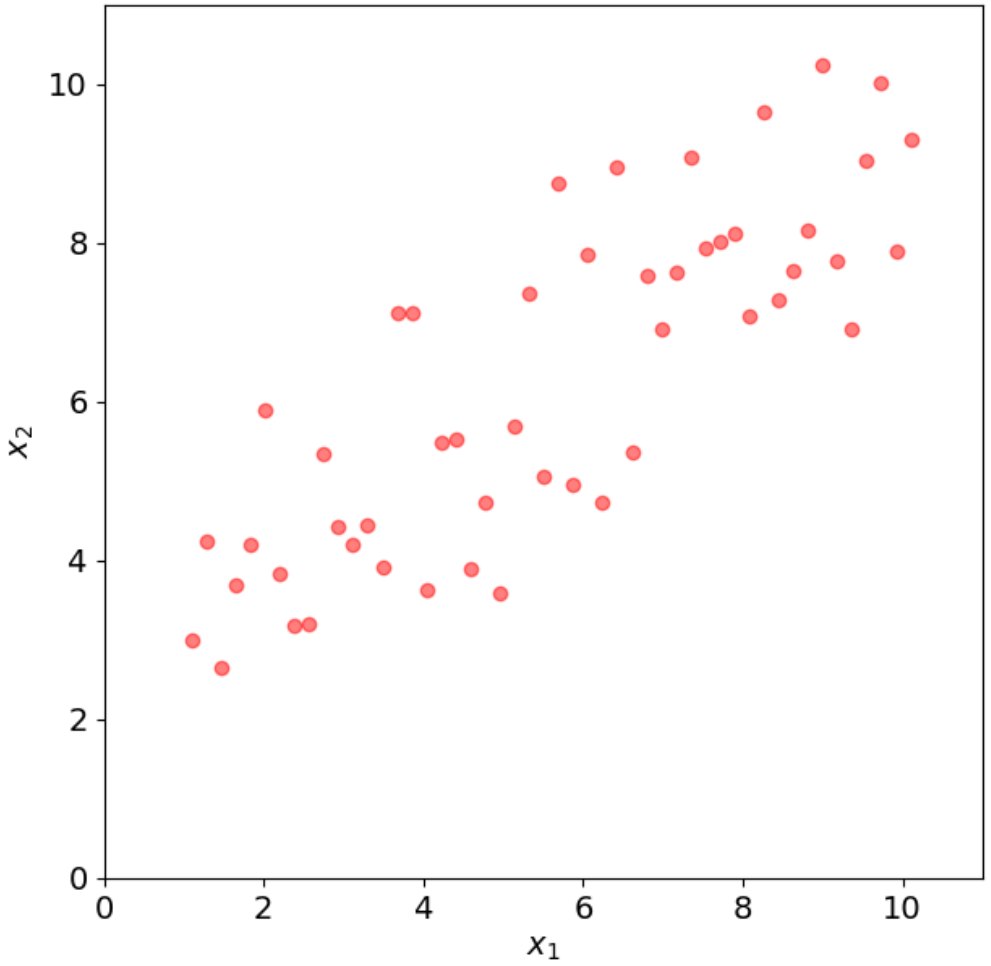
$\{ E_1, E_2, \ldots, E_K \}$

$E_i = f(\{X\})$

where $K < N$

Principal Components Analysis (PCA)

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



- Center data:

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

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

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

- Calculate eigenvalue decomposition:

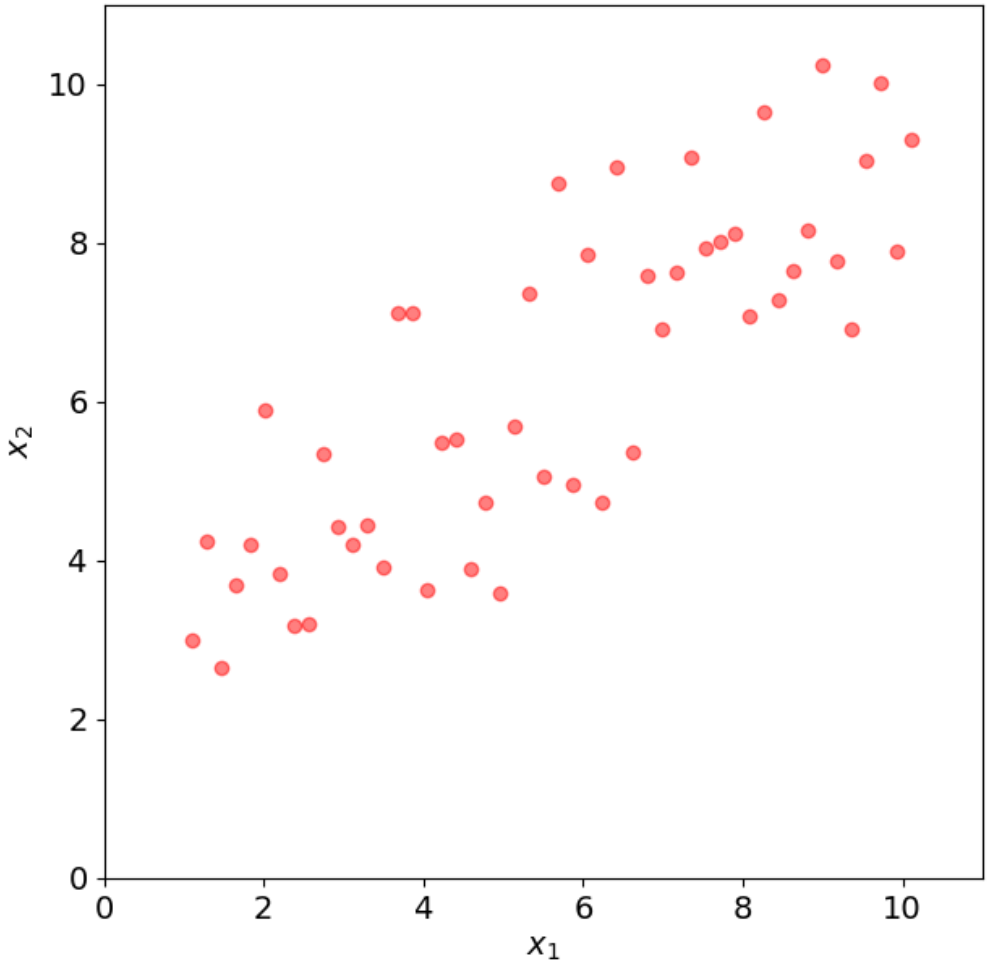
$$\boldsymbol{C} = \boldsymbol{V}\boldsymbol{\lambda}\boldsymbol{V}^{-1}$$

*N*x*N* matrix of ***eigenvectors***

*N*x*N* diagonal matrix of ***eigenvalues***

[Extra] Principal Components Analysis (PCA)

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



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

$$\boldsymbol{Cv} = \boldsymbol{\lambda v}$$

Find eigenvalues as the roots of the characteristic polynomial:

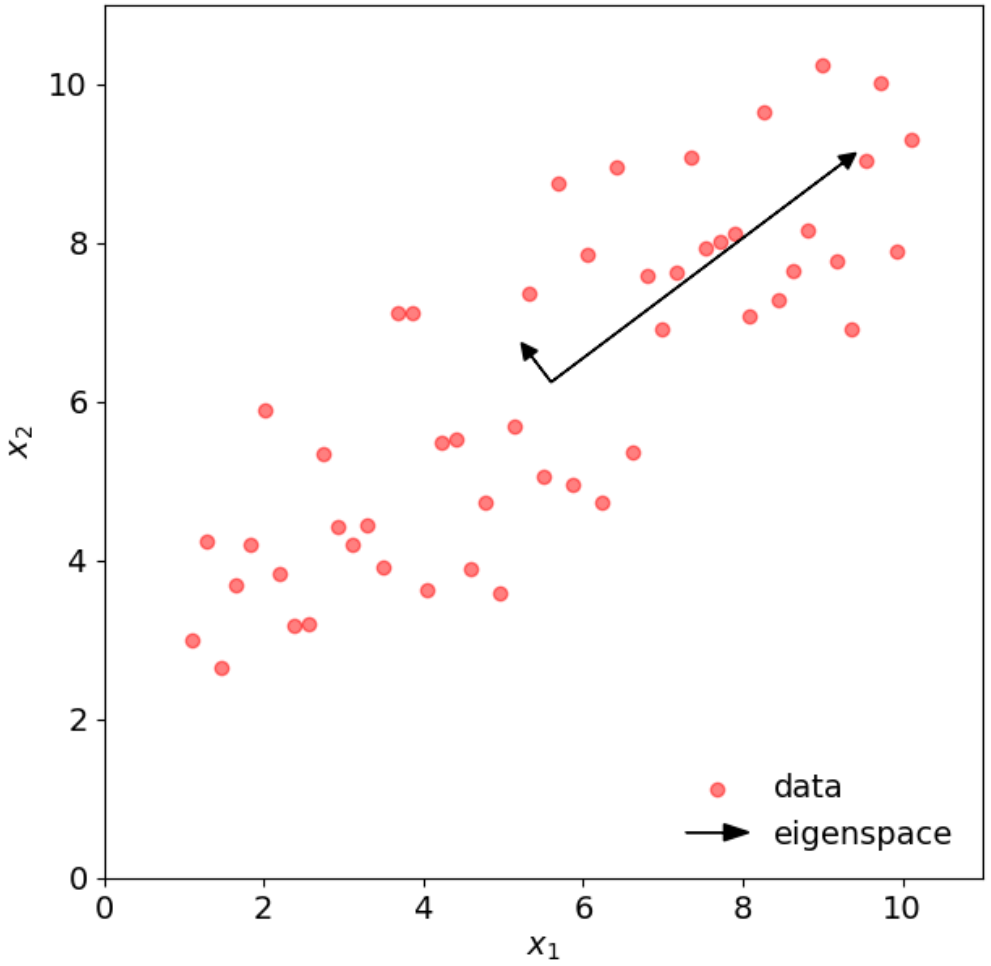
$$p(\boldsymbol{\lambda}) = \det(\boldsymbol{C} - \boldsymbol{\lambda I}) = 0$$

The i-th eigenvector $\boldsymbol{v_i}$ is found by solving:

$$\boldsymbol{Cv_i} = \boldsymbol{\lambda_i v_i}$$

Principal Components Analysis (PCA)

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



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

$\boldsymbol{V = [v_1 \dots v_N]}$ eigenvectors: orthonormal base

$\boldsymbol{\lambda}$ eigenvalues: scalars defining the importance of each eigenvector

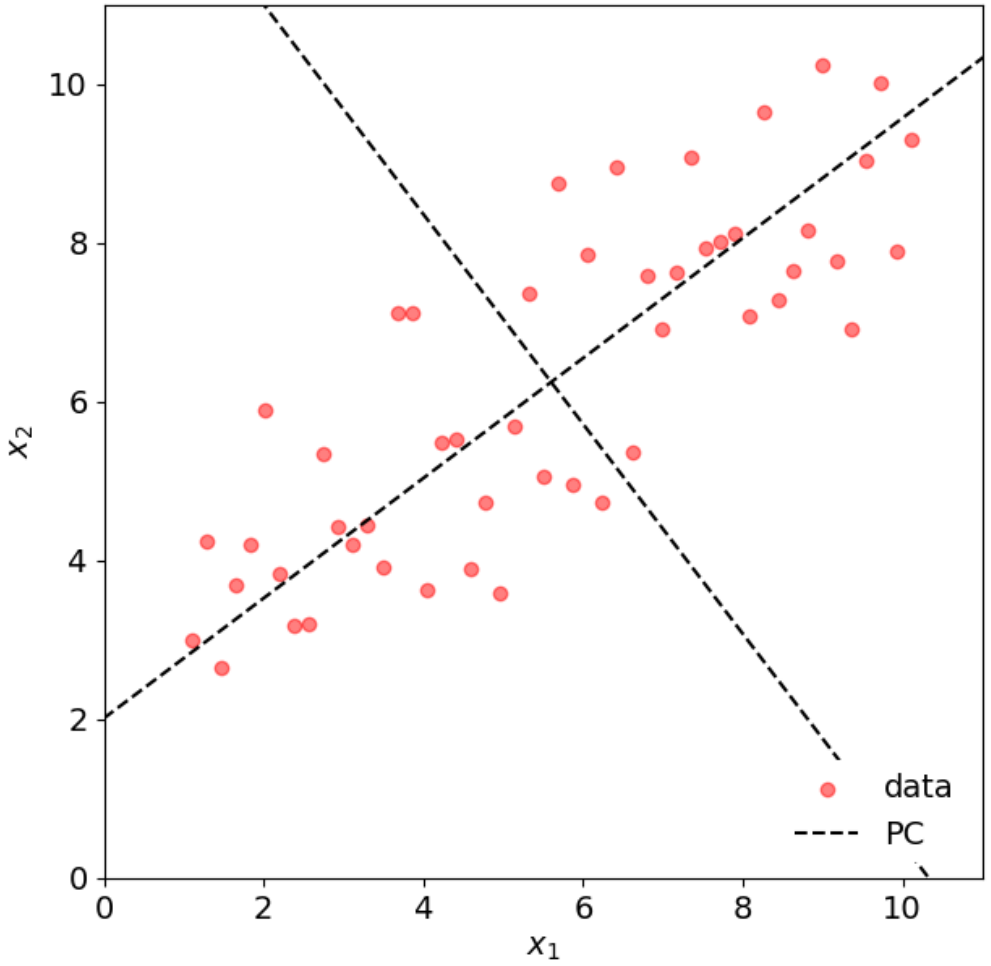
Importance $\boldsymbol{r_i}$ of each eigenvector $\boldsymbol{v_i}$:

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

Sort $\boldsymbol{V}$ and $\boldsymbol{\lambda}$ according to $\boldsymbol{\lambda}$

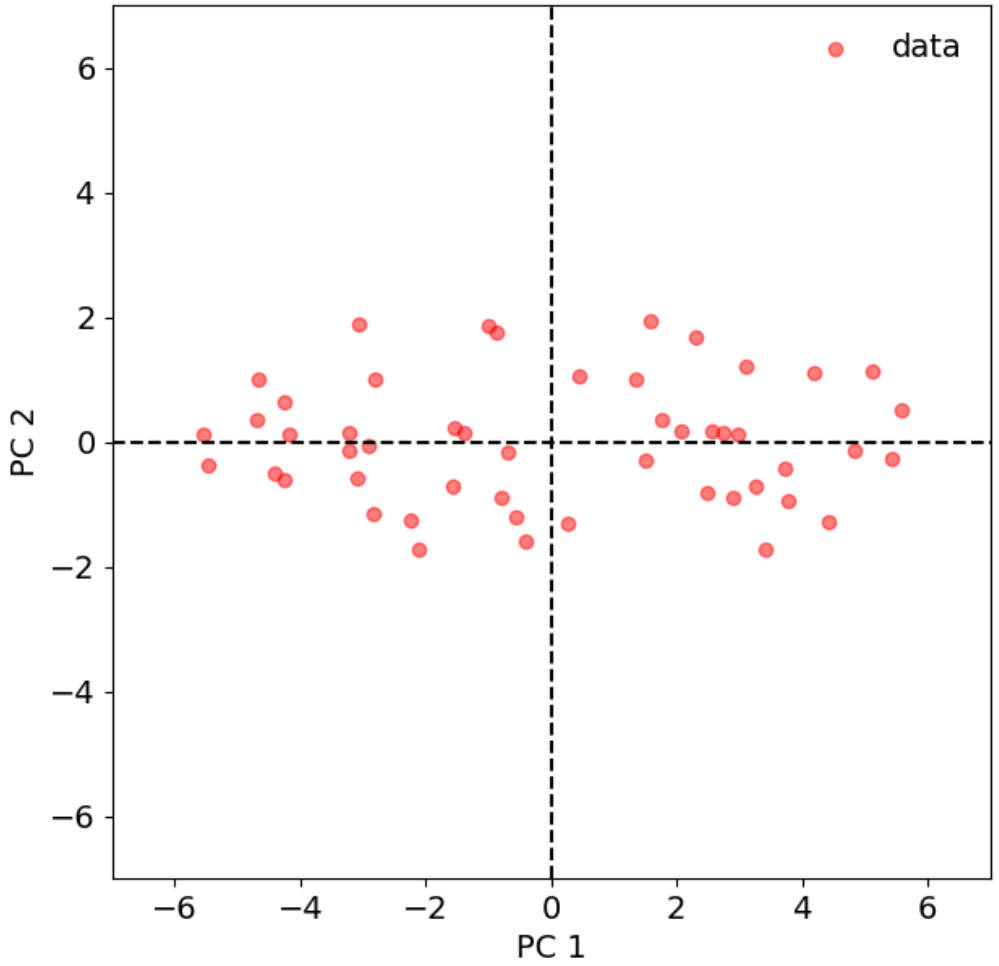
Projection into the eigenspace

Let **X** a dataset of *M* datapoints in *N* dimensions (here, *M*=50 and *N*=2)



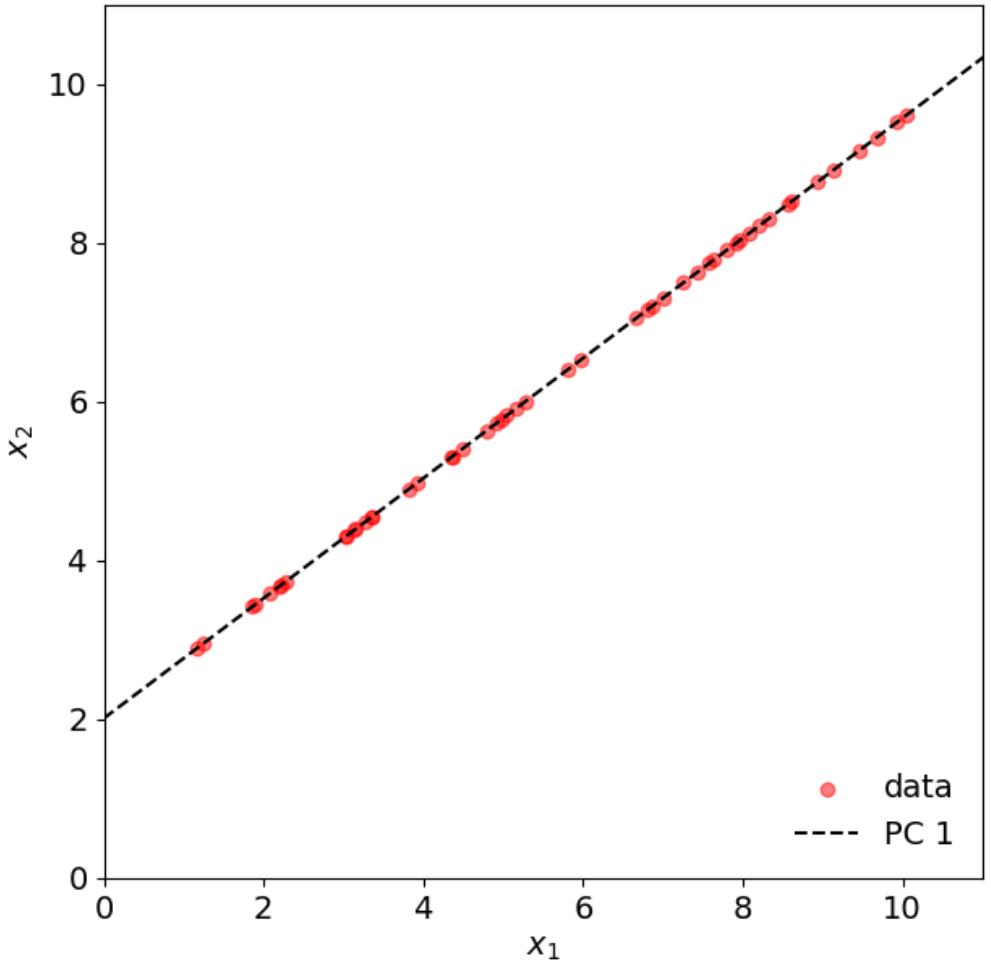
$$p = \boldsymbol{V}^T(\boldsymbol{x} - \mu)$$

transform



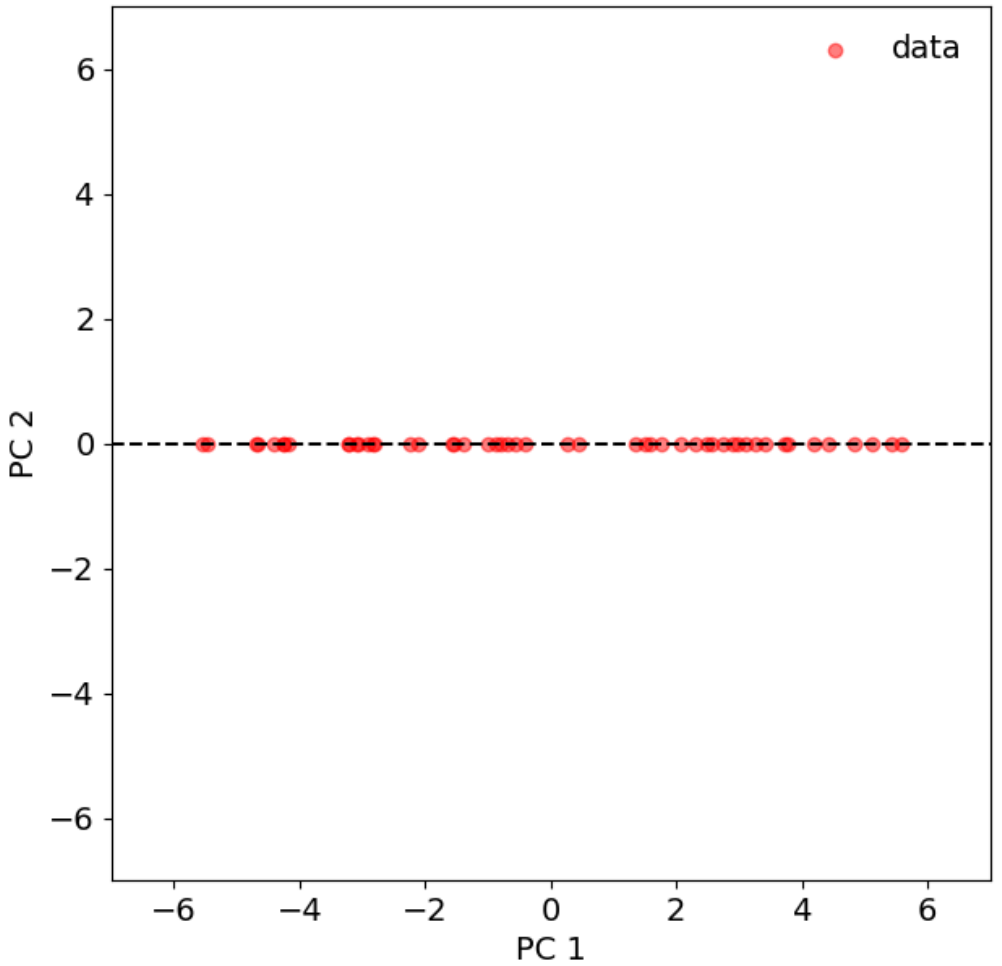
Dimensionality reduction

Remove dimensions that least contribute to data variance



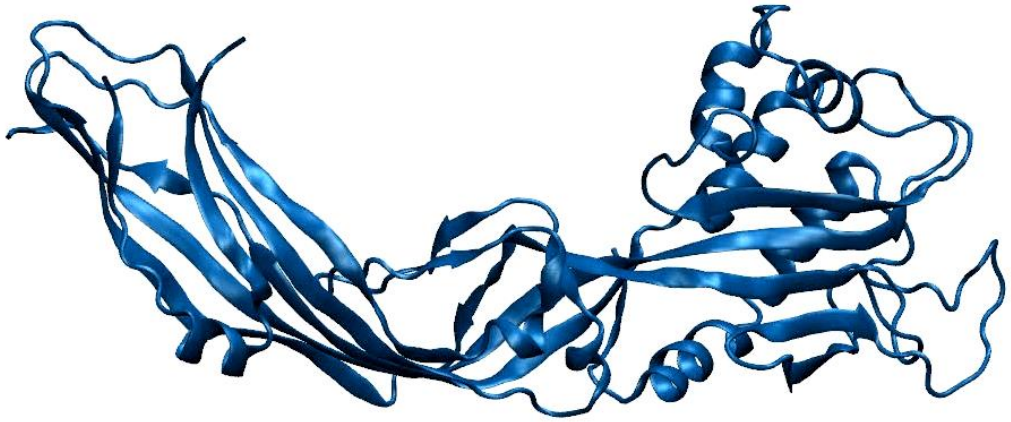
$$p = \boldsymbol{V}^T(\boldsymbol{x} - \mu)$$

transform

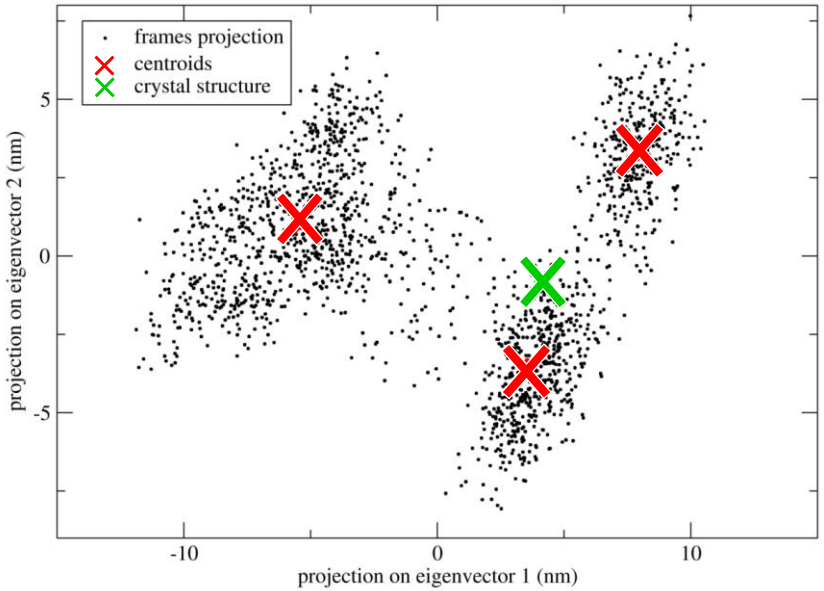


[Example] Identifying dominant motions in proteins

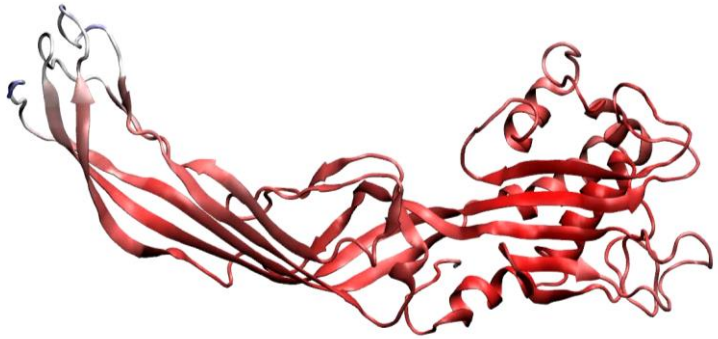
Protein MD simulation



Eigenspace of Cα coordinates



- Simulations are complex and noisy
- select only first few PC (eigenvectors) to separate signal from noise



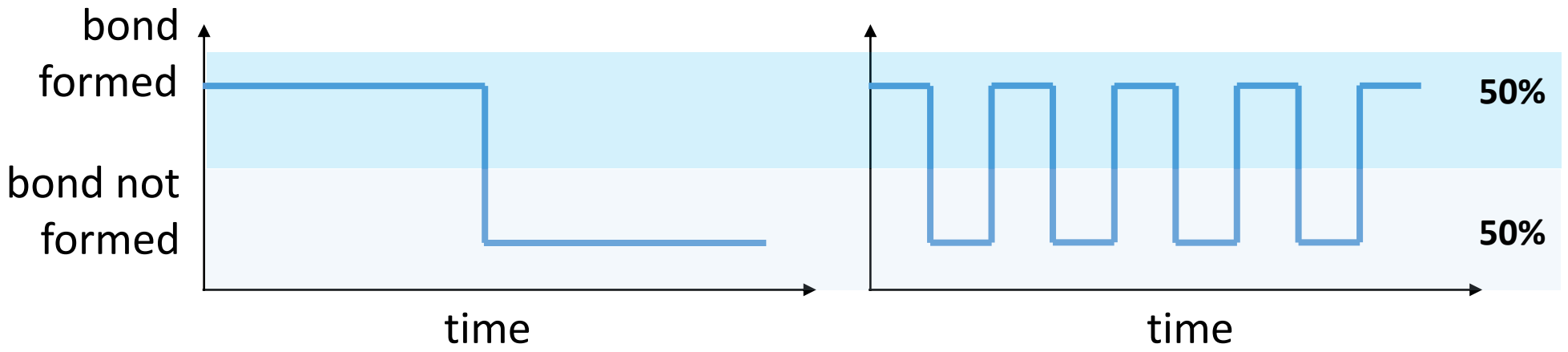
Degiacomi, M.T. & Dal Peraro, M.. *Macromolecular symmetric assembly prediction using swarm intelligence dynamic modeling*. ***Structure***, 2013

Time-lagged independent component analysis (tICA)

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

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

Thought experiment: typically hydrogen bond is considered established if donor-acceptor distance <2.5 Å, and donor-acceptor-hydrogen angle <20°.



reporting % time a bond is established in simulation can be misleading!

Time-lagged independent component analysis (tICA)

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

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

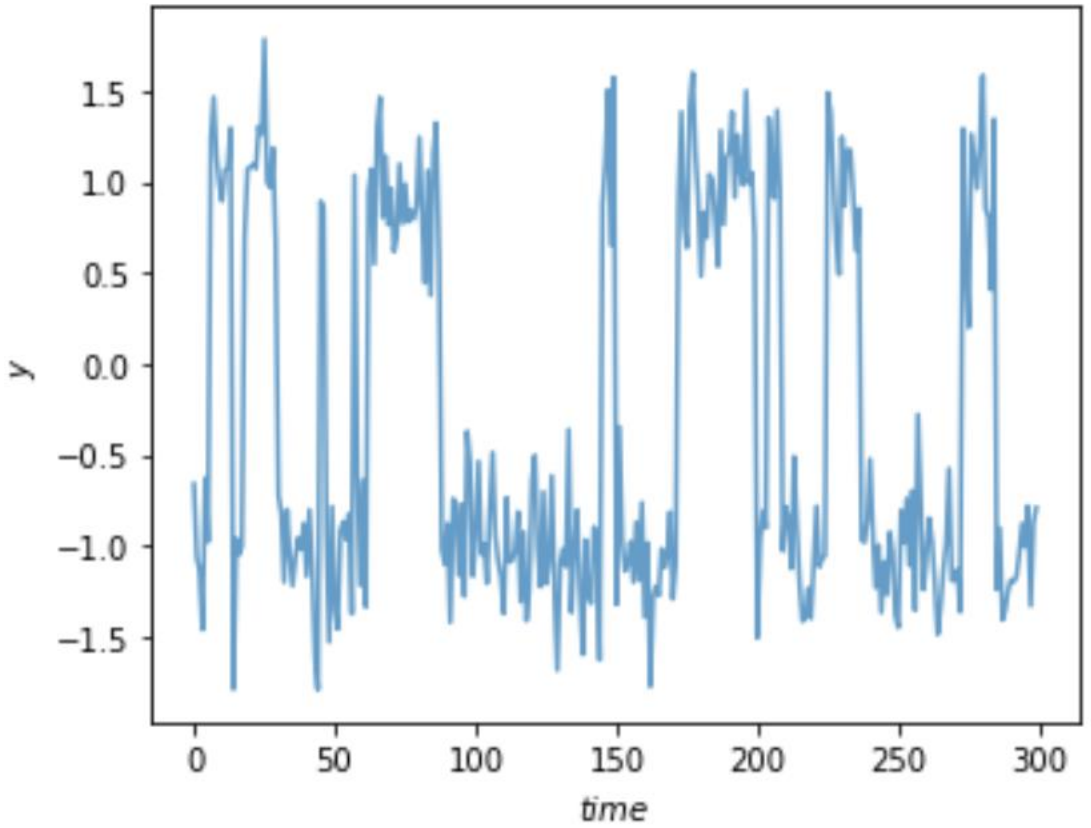
$\mathbf{r}(t) = (r_i(t))_{i=1,\ldots,D}$

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 $\mathbf{C}(\tau)$



F. Nüske, […], A. S. J. S. Mey and F. Noé: *J***CTC**, 2014

G. Perez-Hernandez, et al., ***JCTC***, 2013

C. R. Schwantes et al., ***JCTC***, 2013

Time-lagged independent component analysis (tICA)

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

The transform is chosen such that, amongst all linear transforms, tICA **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:

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

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

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

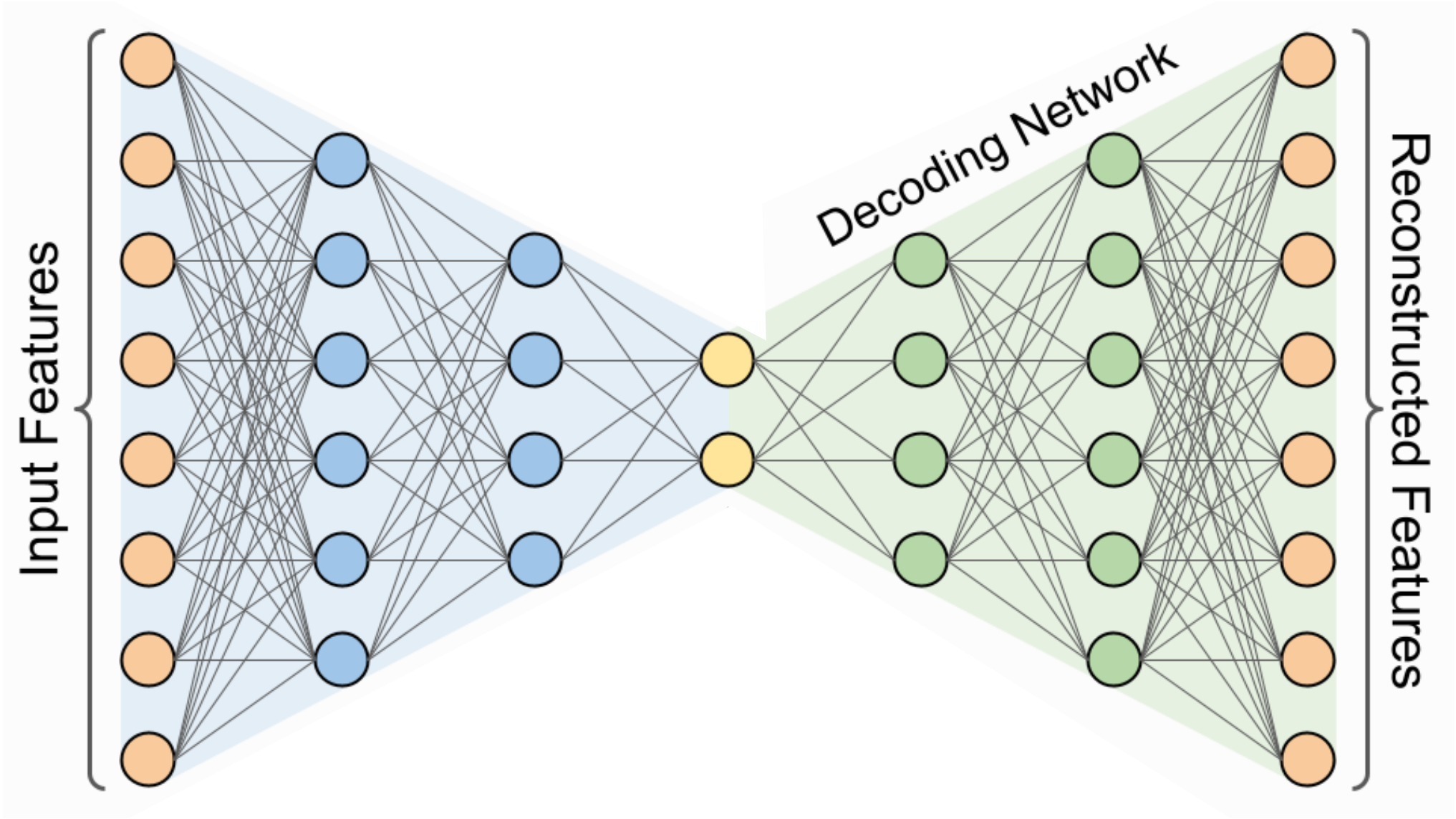
M columns of full rank U for DR

F. Nüske, […], A. S. J. S. Mey and F. Noé: *JCTC*, 2014

G. Perez-Hernandez, et al., ***JCTC***, 2013

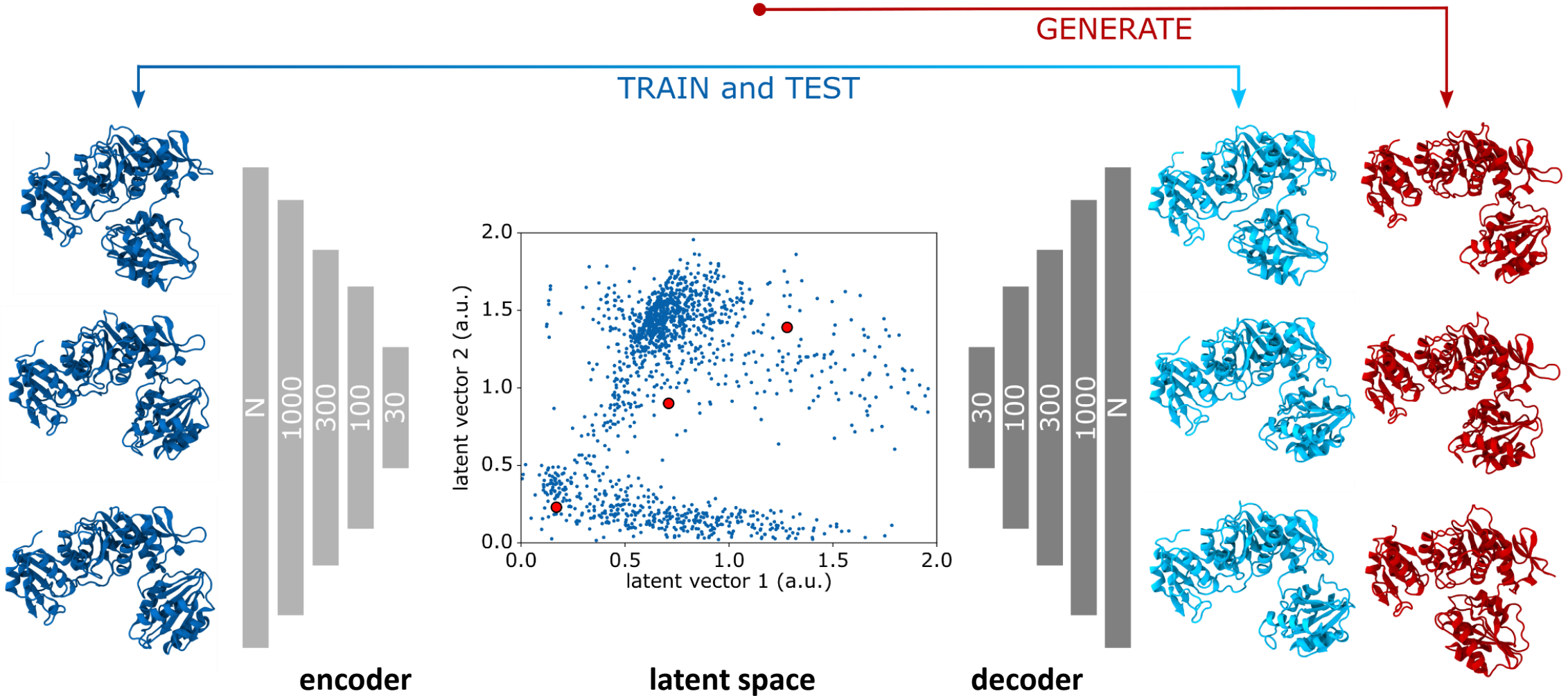
C. R. Schwantes et al., ***JCTC***, 2013

[Extra] Dimensionality reduction with neural networks



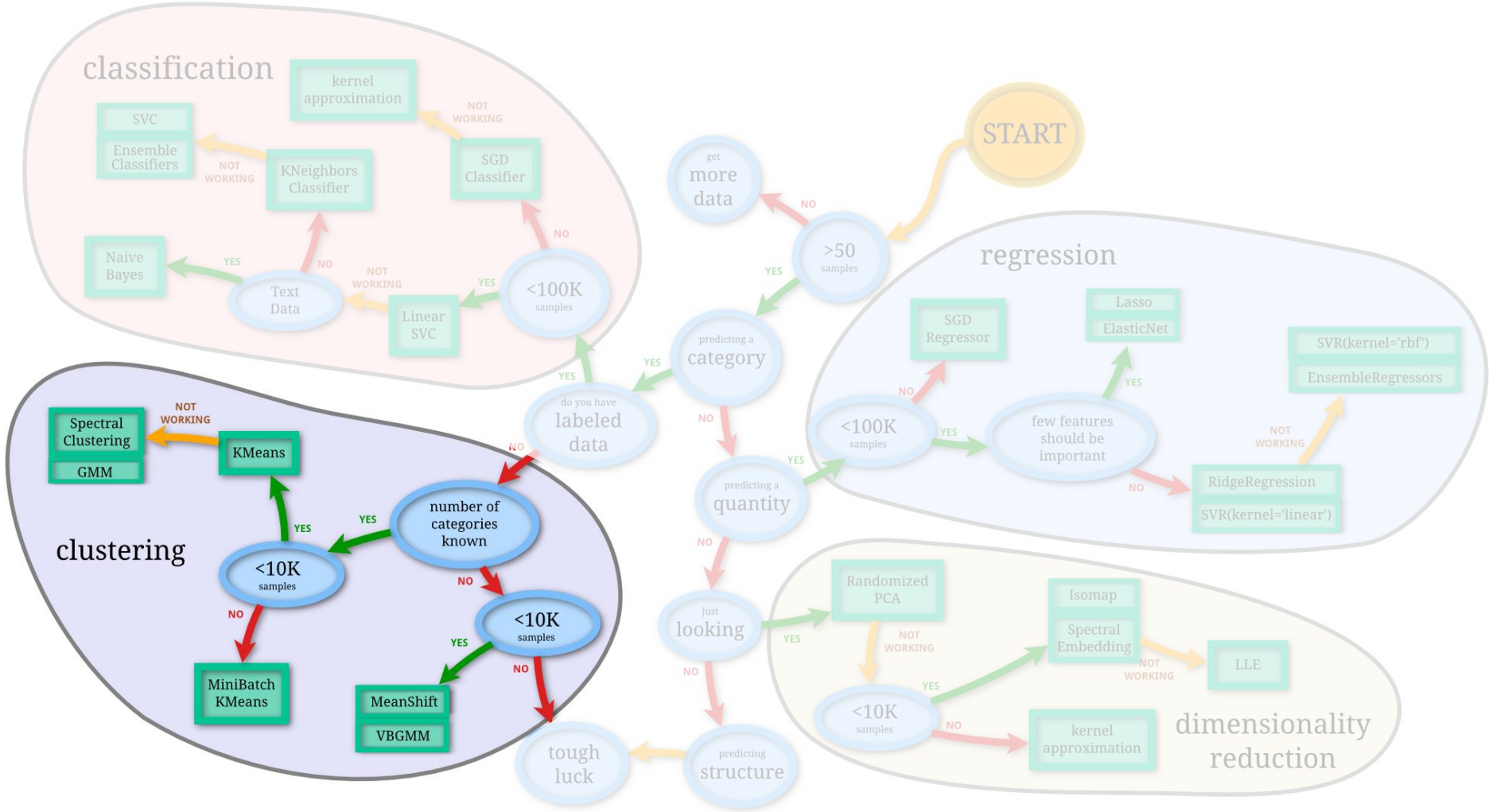
PCA dimensionality reduction is *interpretable*, that of t-SNE and neural networks is not.

[Extra] Learning Protein Conformational Space



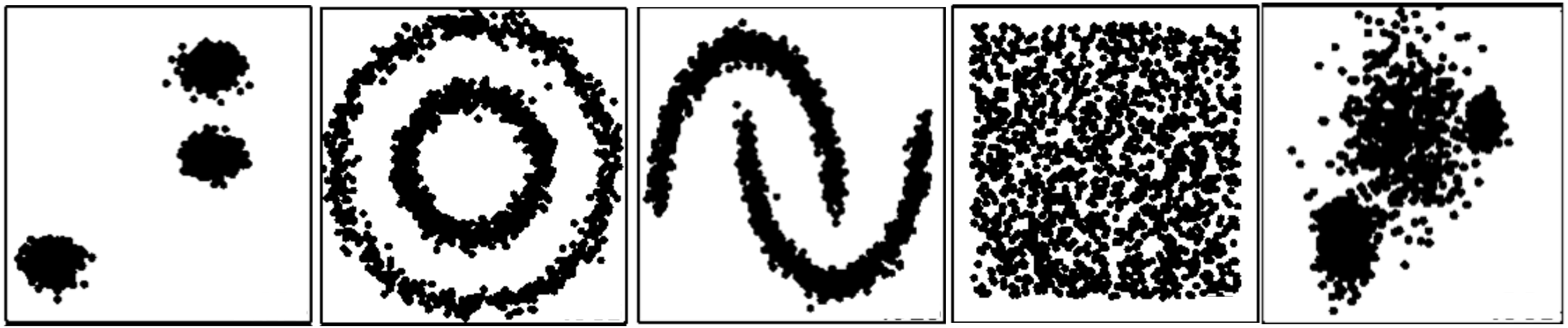
M.T. Degiacomi, ***Structure***, 2019

The Data Mining world



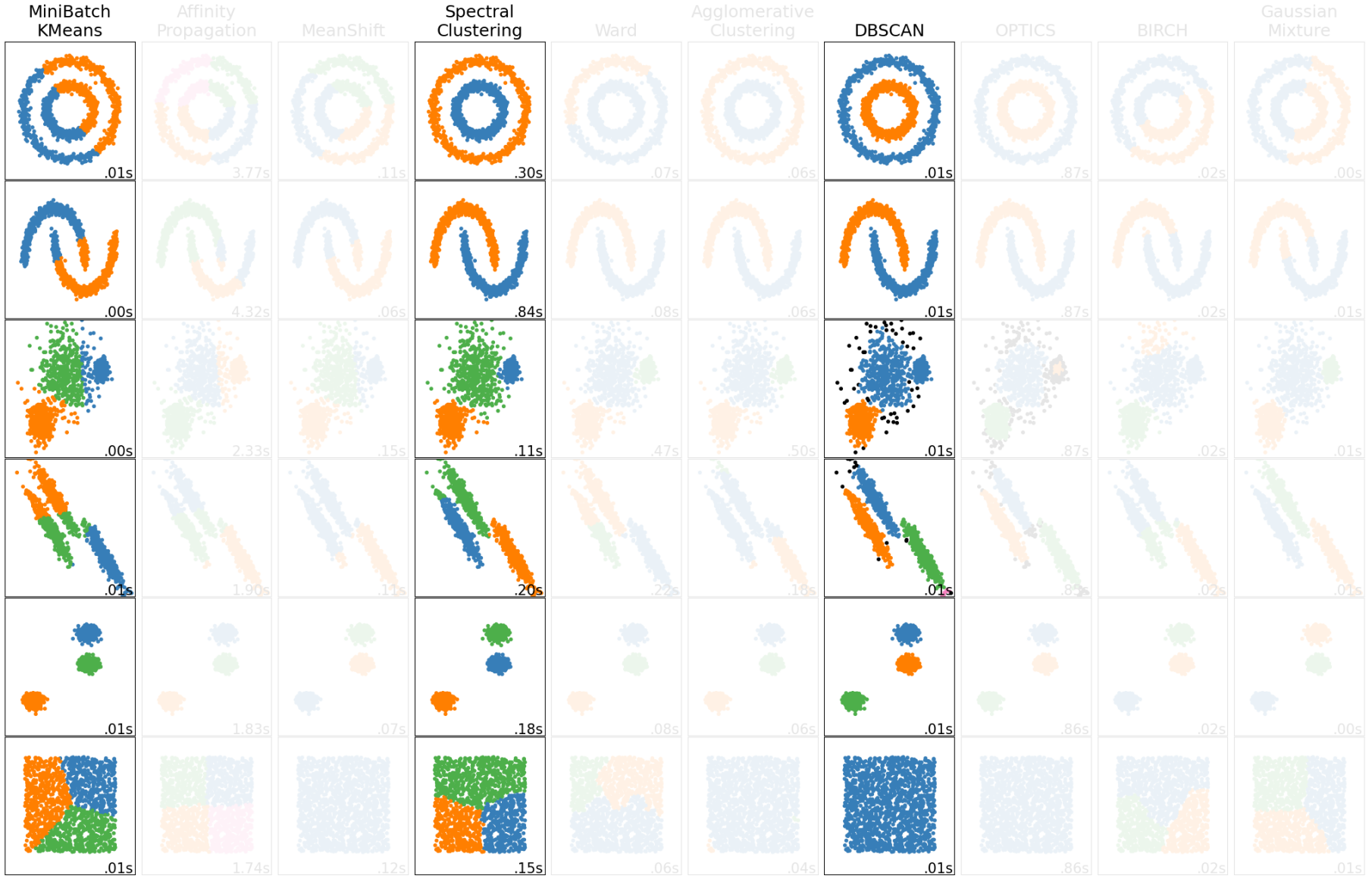
From **scikit-learn.org**

Clustering (i.e., unsupervised learning)



Known number of clusters? Flat geometry?
Even cluster size? Outliers? Centroids needed?

Clustering algorithms



scikit-learn.org/stable/modules/clustering.html

How does k-means work?

Input: K, set of points $x_1 \ldots x_n$ (can be in N-dimensional)

Place centroids, $c_1 \ldots, c_n$ at random locations

Repeat until convergence:

For each point x_i:

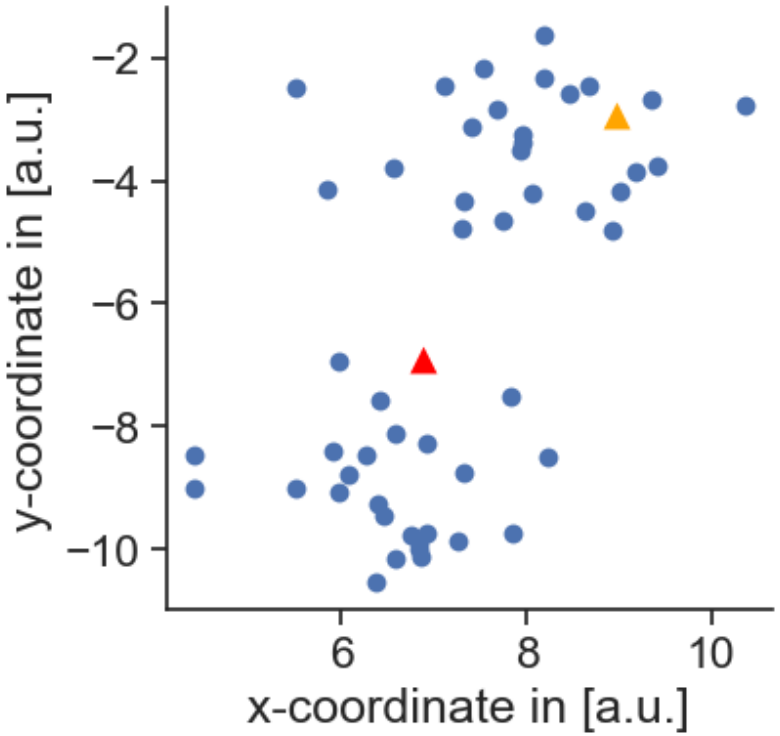
Find nearest centroid $c_j = \arg\min_j D(x_i, c_j)$

Assign the point x_i to cluster j

For each cluster j = 1…K:

Compute the centroid mean for all points in one cluster and update the centroid

$$c_j(a) = \frac{1}{n_j} \sum_{x_i \to c_j} x_i(a)$$



How does k-means work?

Input: K, set of points $x_1 \ldots x_n$ (can be in N-dimensional)

Place centroids, $c_1 \ldots, c_n$ at random locations

Repeat until convergence:

For each point x_i:

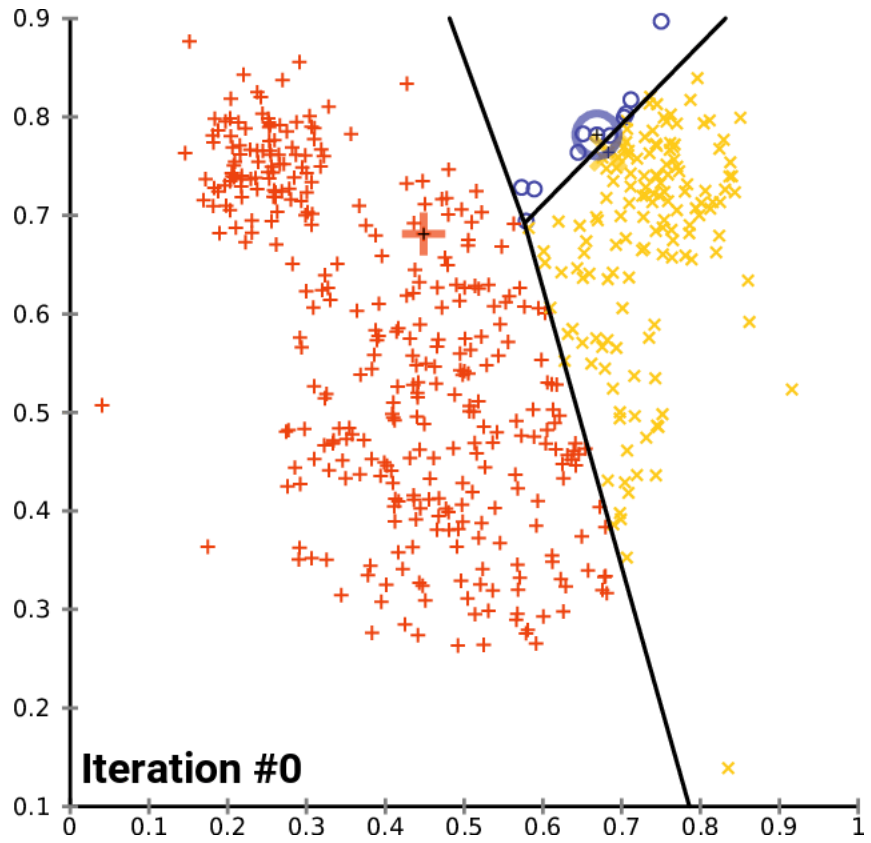
Find nearest centroid $c_j = \arg\min_j D(x_i, c_j)$

Assign the point x_i to cluster j

For each cluster j = 1…K:

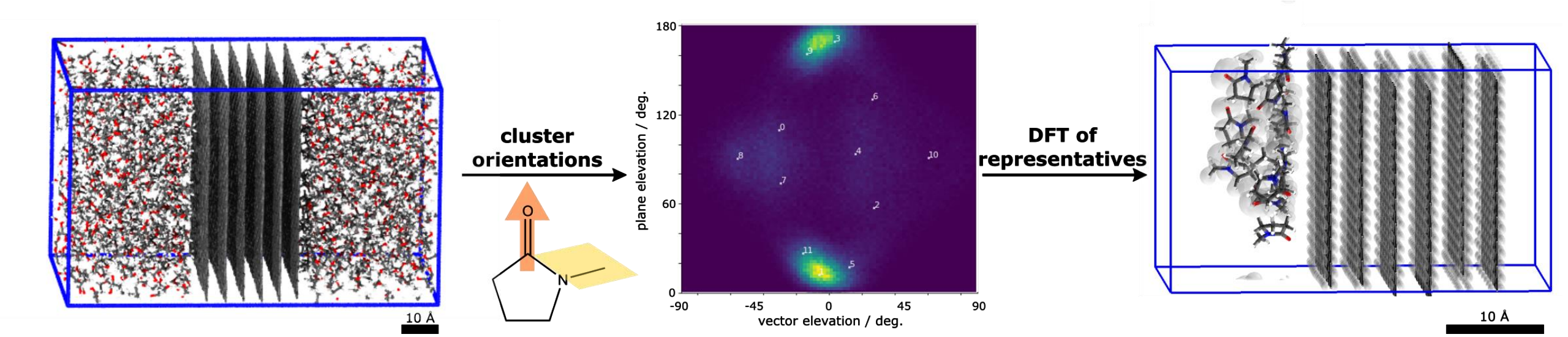
Compute the centroid mean for all points in one cluster and update the centroid

$$c_j(a) = \frac{1}{n_j} \sum_{x_i \to c_j} x_i(a)$$



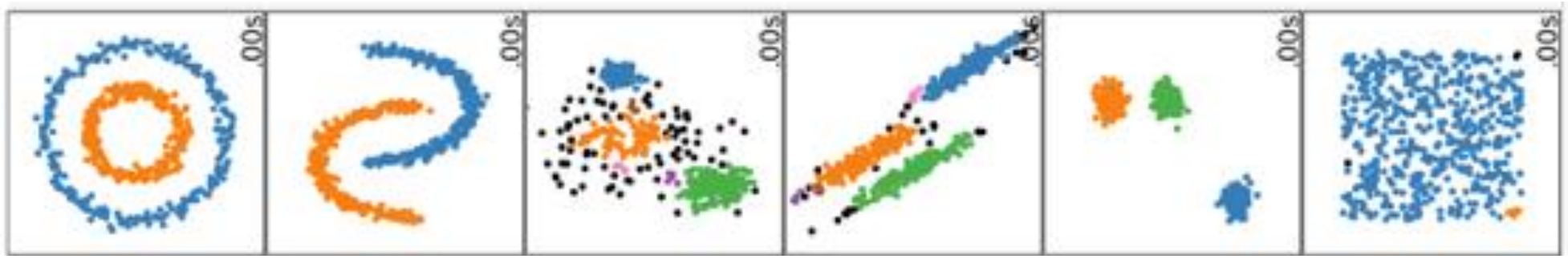
[Example] k-means vs solvent-graphite interactions

- Molecular Dynamics simulation of graphite immersed in solvents.
- >100k individual solvent-graphene interactions

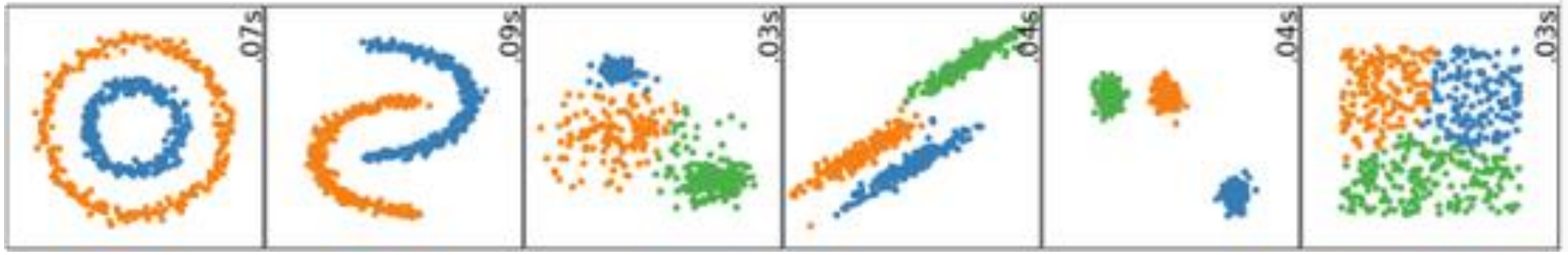


Density-based and spectral clustering

DBSCAN

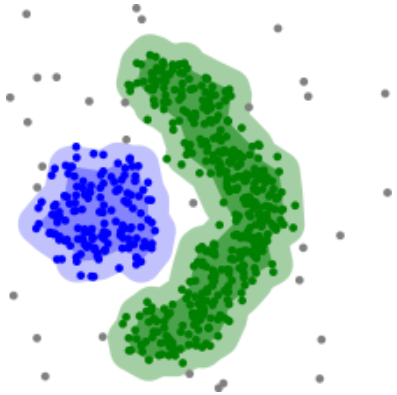


Spectral Clustering



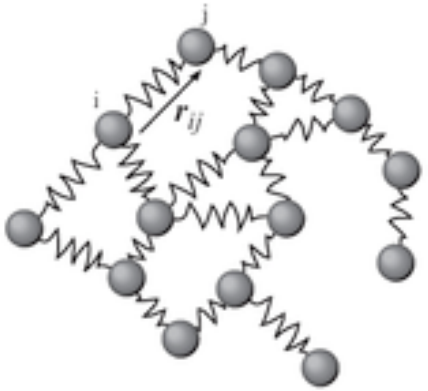
Density-based and spectral clustering

DBSCAN



- Find the points in the ε neighbourhood of every point, identify core points with more than n neighbours.
- Find the connected components of core points on the neighbour graph, ignoring all non-core points.
- Assign each non-core point to a nearby cluster if the cluster is an ε neighbour, otherwise assign to noise otherwise.

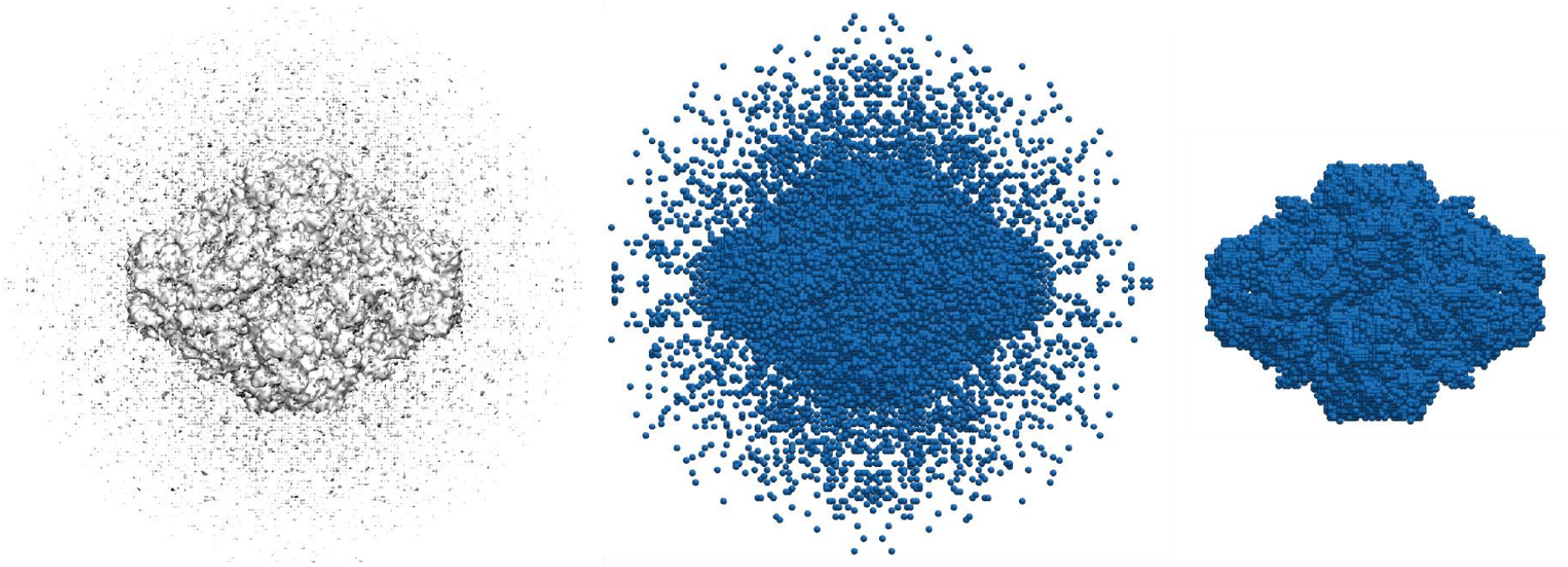
Spectral Clustering



- Calculate the Laplacian
- Calculate the first k eigenvectors
- Consider the matrix formed by the first k-eigenvectors
- Cluster the graph nodes based on these features (e.g. k-means)

[Example] DBSCAN for noise detection in EM maps

- Load electron density map and chose a threshold *t*
- Place pseudoatoms where intensity > *t*
- Cluster beads and delete small clusters (<1% of total beads)



M.T.Degiacomi and J.L.P. Benesch, *EM∩IM: software for relating ion mobility mass spectrometry and electron microscopy data*, ***Analyst***, 2015

Post-its


Break Time!