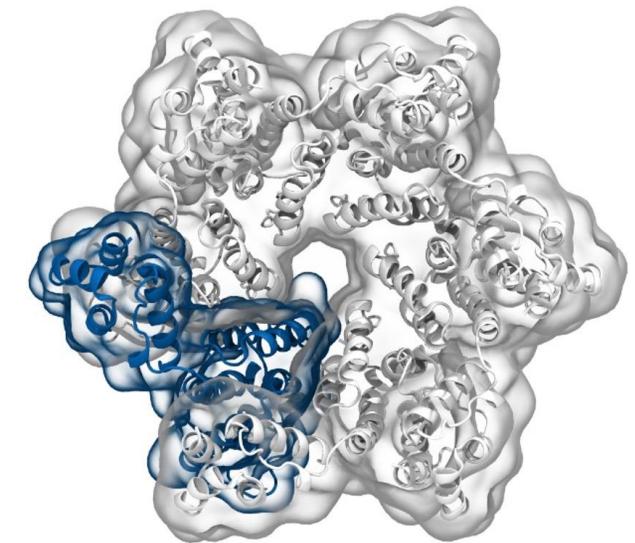
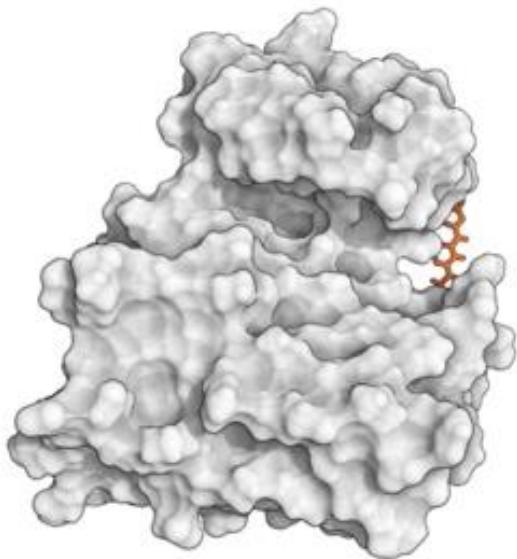


Simulation of Biomolecules

Simulation Analysis part 1

2023 CCP5 Summer School



Dr Matteo Degiacomi

Durham University

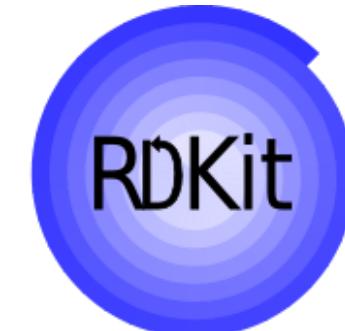
matteo.t.degiacomi@durham.ac.uk

Dr Antonia Mey

University of Edinburgh

antonia.mey@ed.ac.uk

Large ecosystem of Python-based tools data analysis

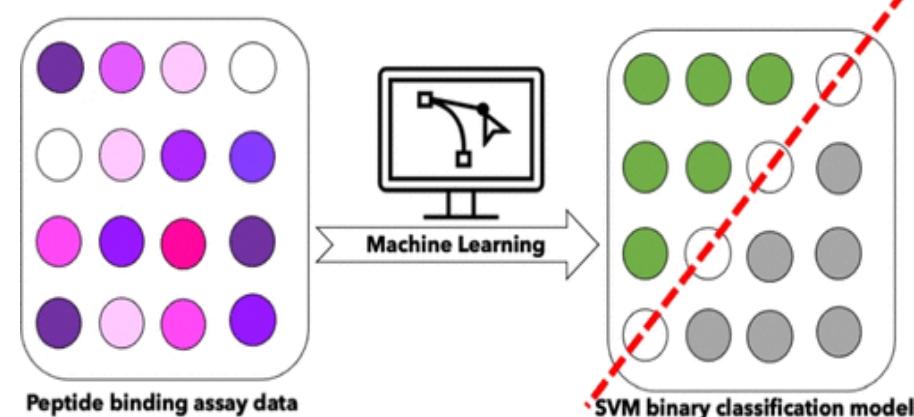
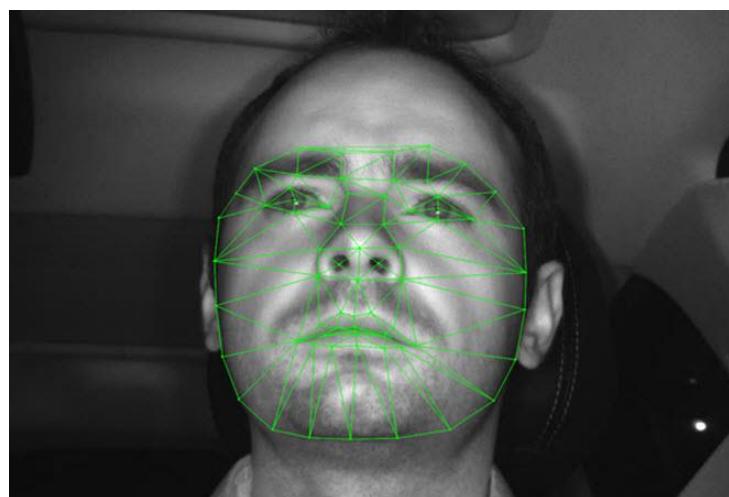
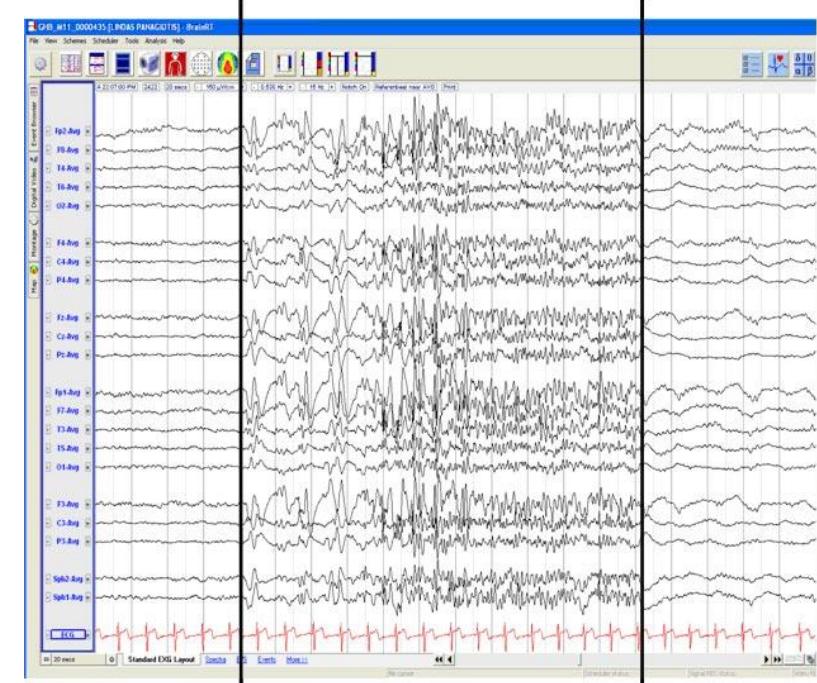
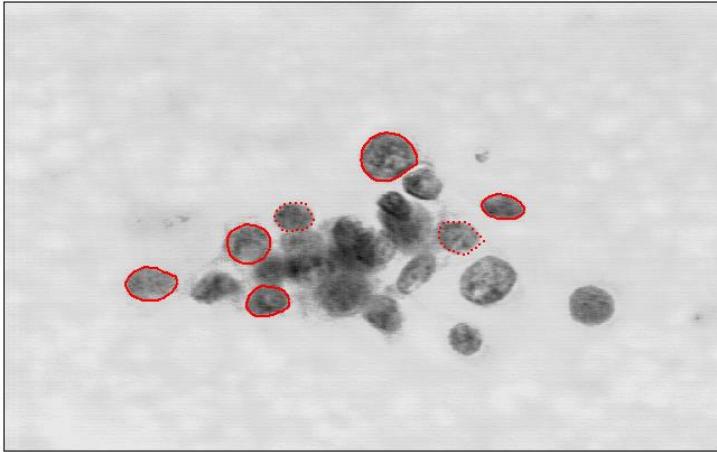


Open-Source Cheminformatics
and Machine Learning

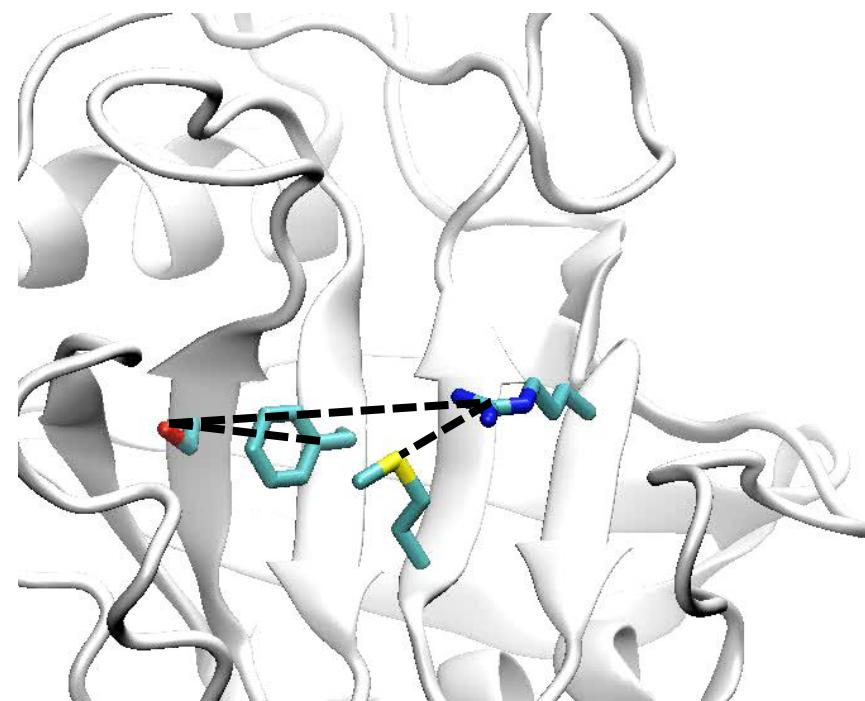
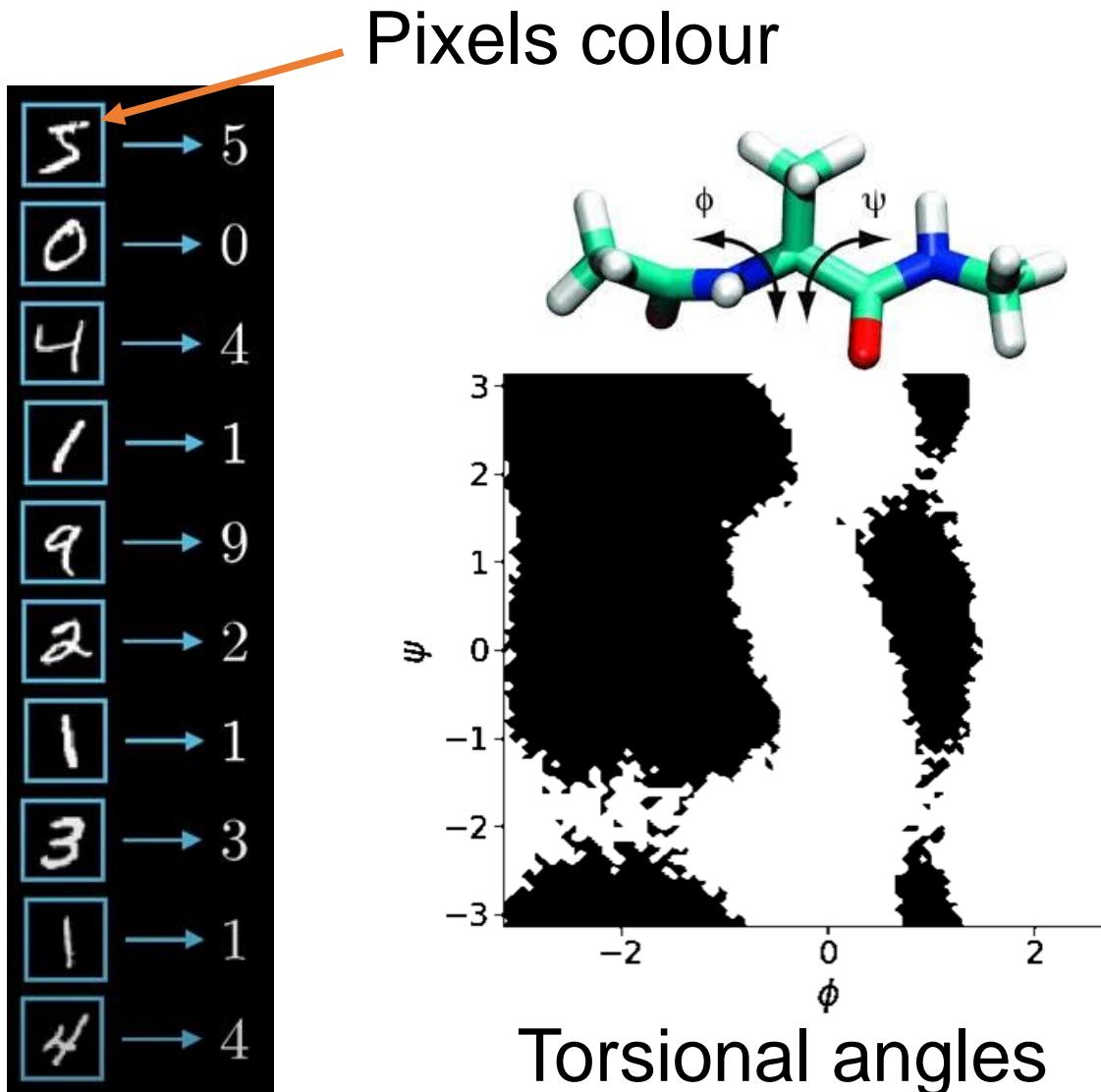
And many more...

MD simulations produce complex, high-dimensional data

S	→ 5
O	→ 0
Y	→ 4
I	→ 1
Q	→ 9
Z	→ 2
L	→ 1
T	→ 3
H	→ 1
N	→ 4

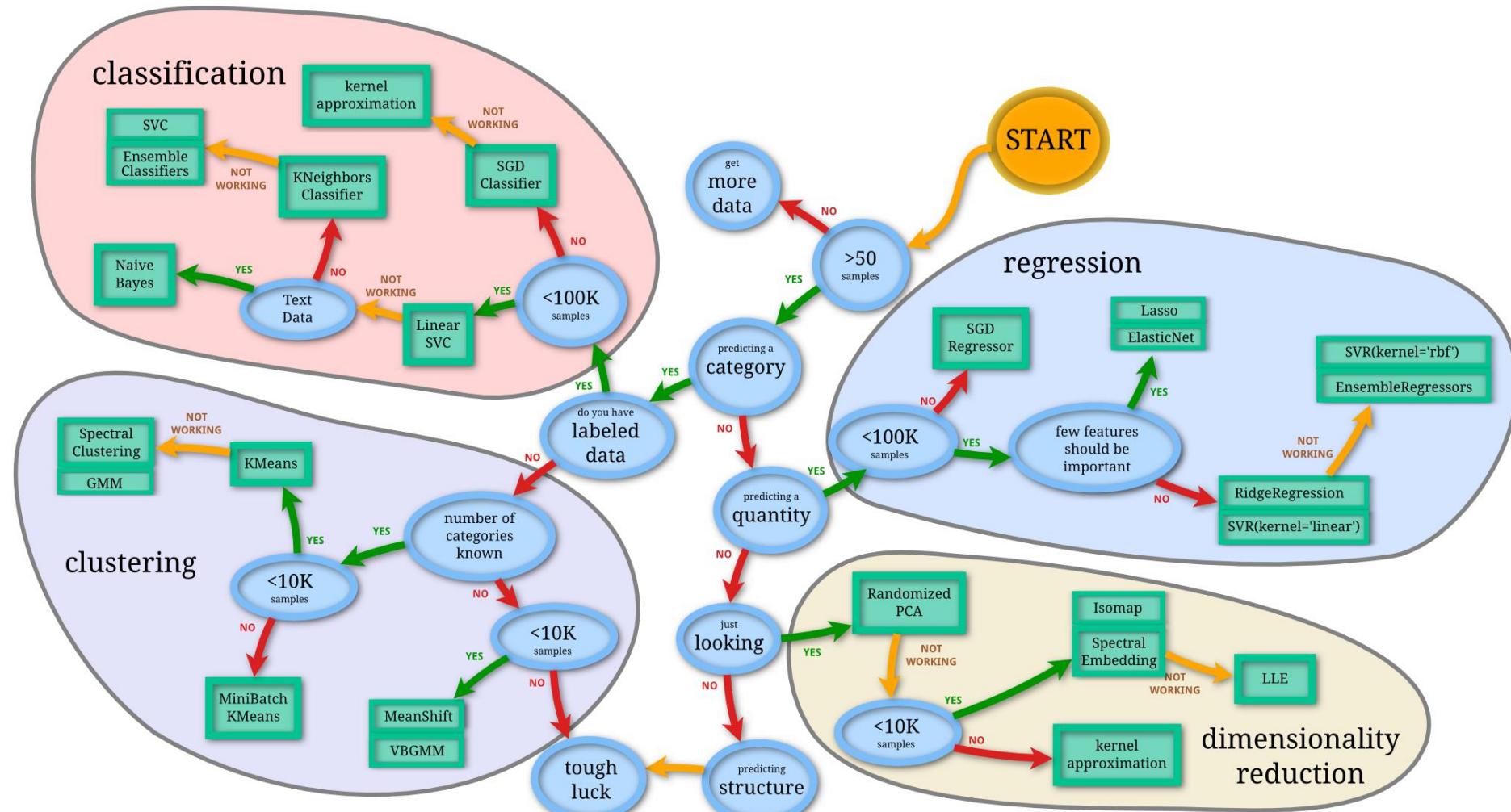


Features are possible ways to represent data

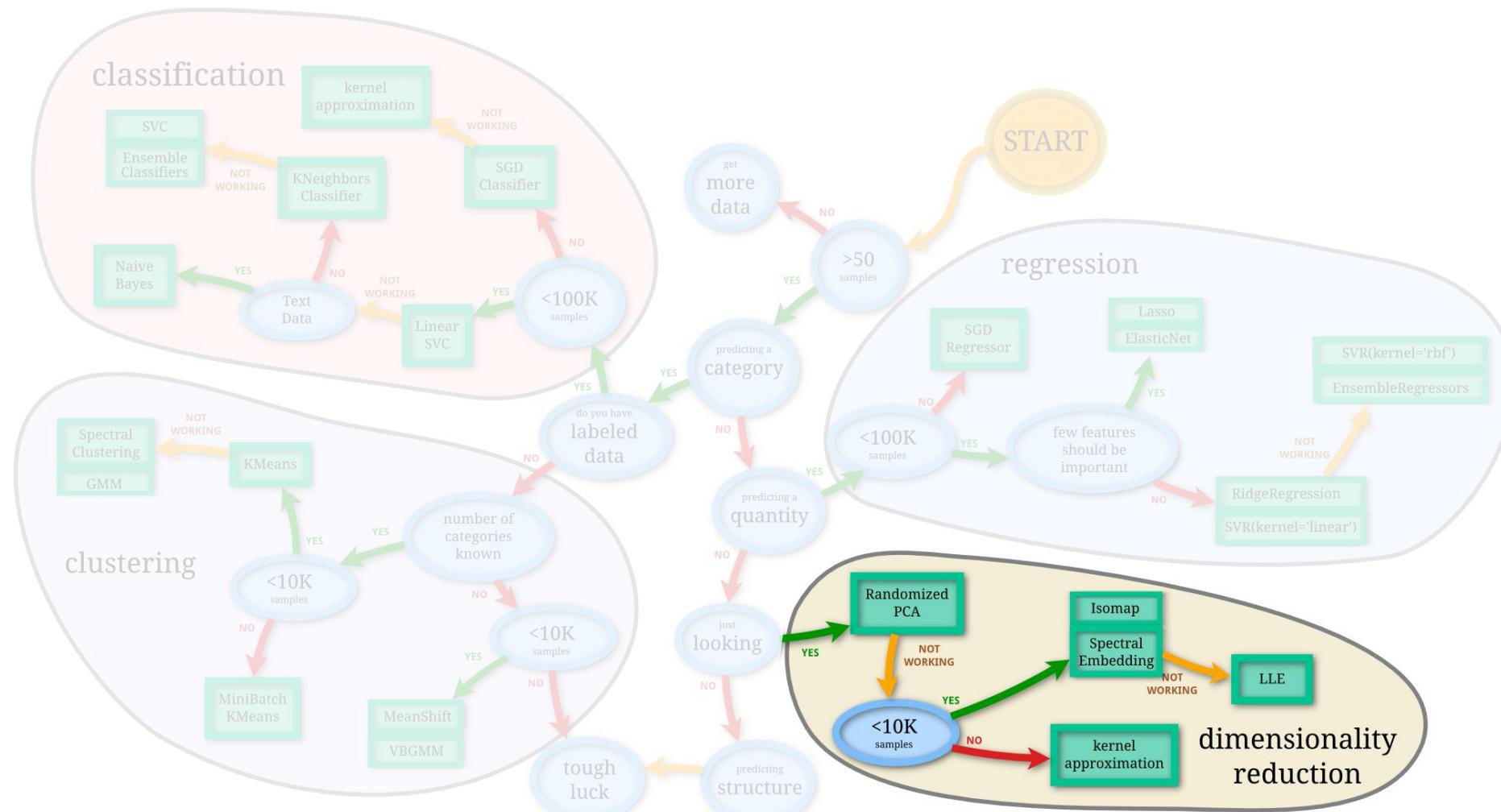


Interatomic distances

The Data Mining world

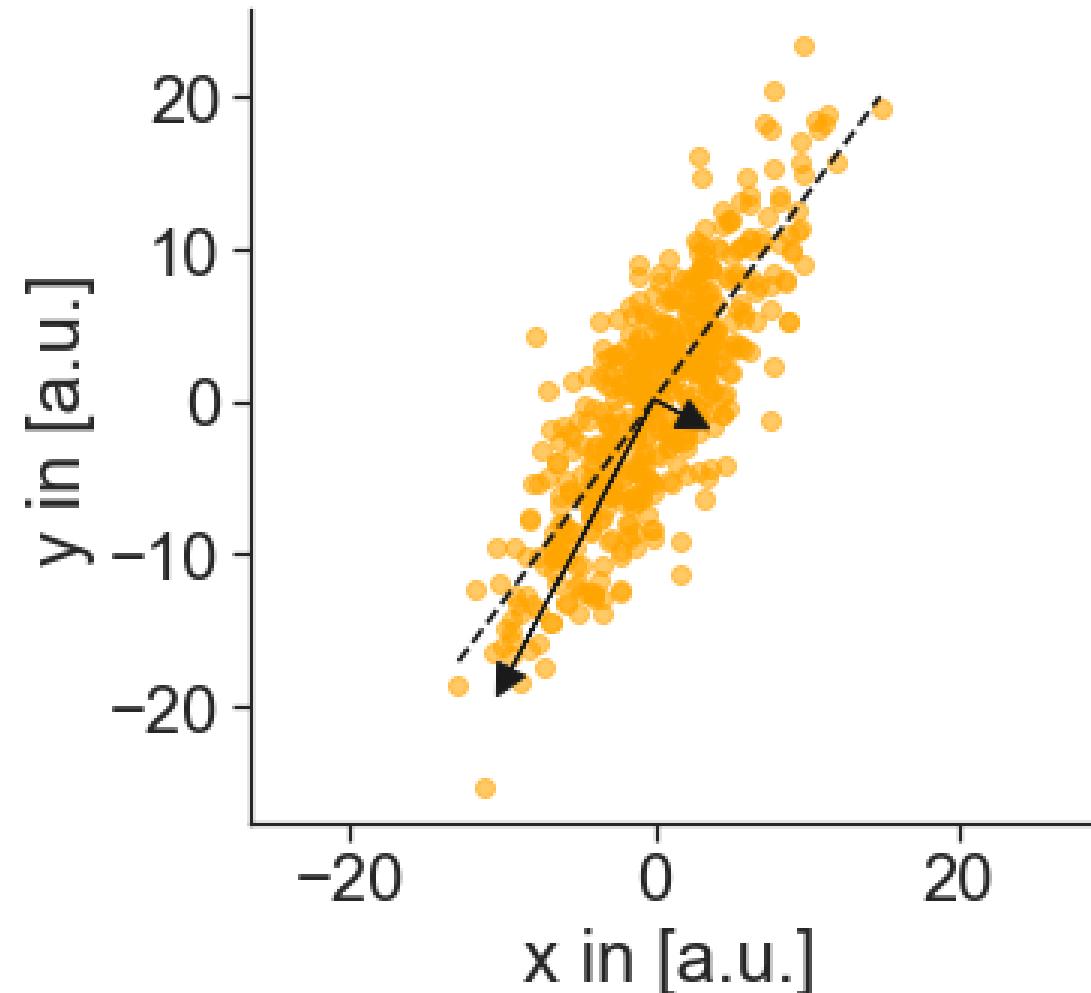


The Data Mining world



Principal Components Analysis (PCA)

- Finds the reference system where each dimension maximises data variance.
- “*Data is mostly spread along the first dimension, then the second, ...*”
- Lowest dimensions describe noise
- First principal component does *not* align with line of best fit



PCA, formally

Let $\mathbf{C}(0)$ the covariance matrix of some data \mathbf{X}

Let the generalised eigenvalue problem:

$$\mathbf{C}(0)\mathbf{W} = \mathbf{W}\Sigma$$

Gives an eigenvector matrix \mathbf{W} that will allow the transform of the original data \mathbf{X} onto a new basis \mathbf{T} that maximises the variance.

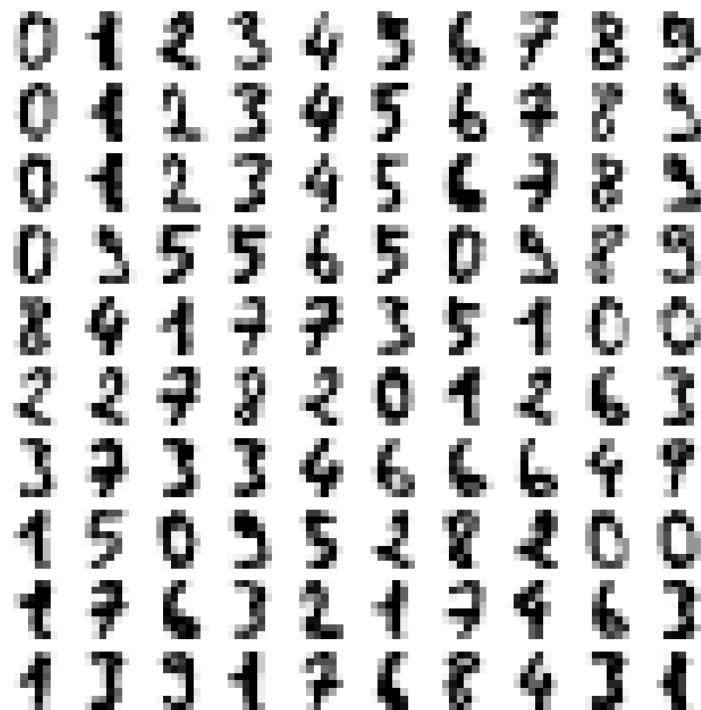
$$\mathbf{T} = \mathbf{X}\mathbf{W}$$

It is possible to choose m eigenvectors to project onto by only using the first m -columns of \mathbf{W} .

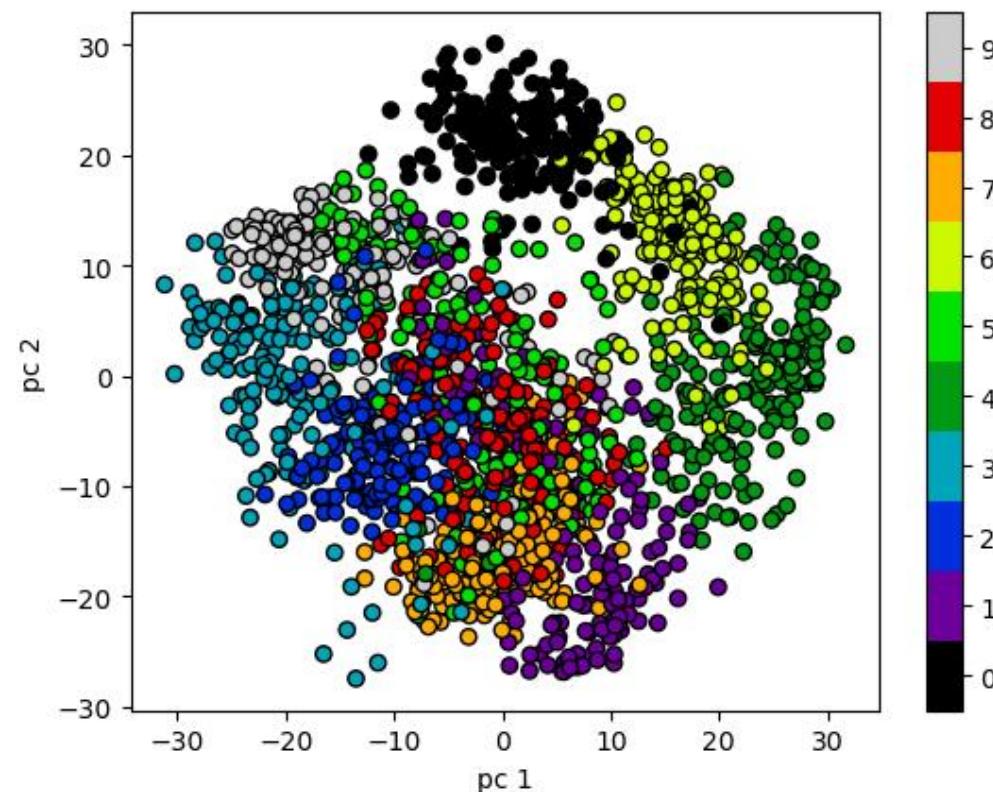
[Example 1] representing written digits

MNIST database of written digits

Input is a vector of 64 dimensions
8x8 pixel digits

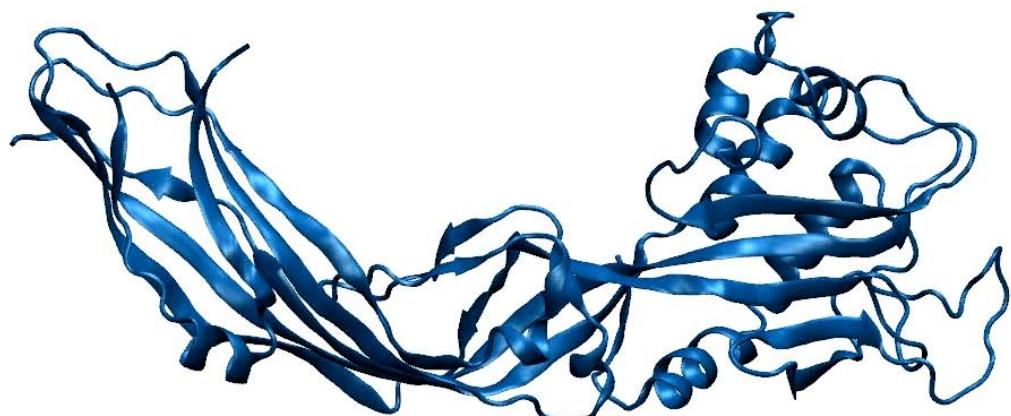


2 principal components manage to project
a few of the digits in a similar area of space!

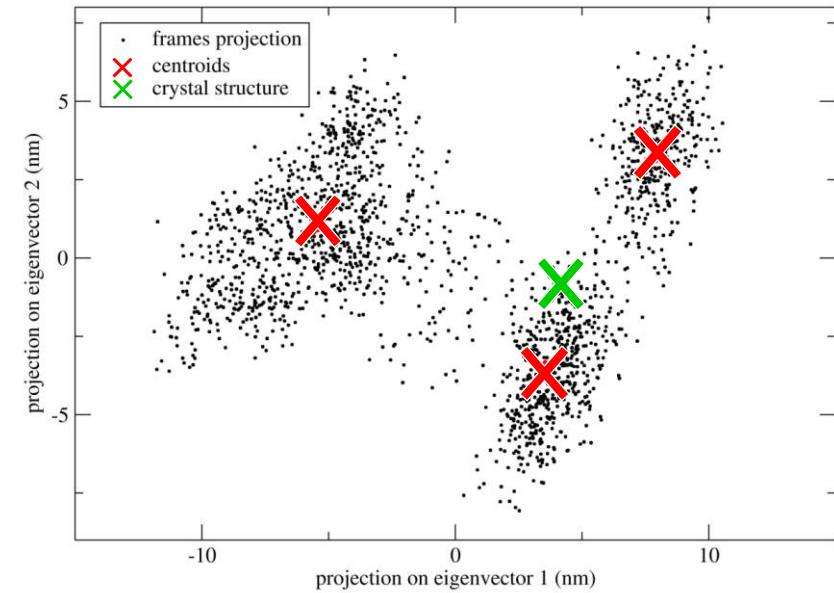


[Example 2] Identifying dominant motions in proteins

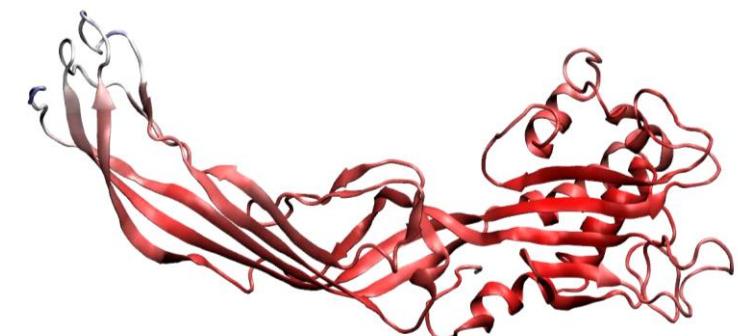
Protein MD simulation



Project MD in eigenspace



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

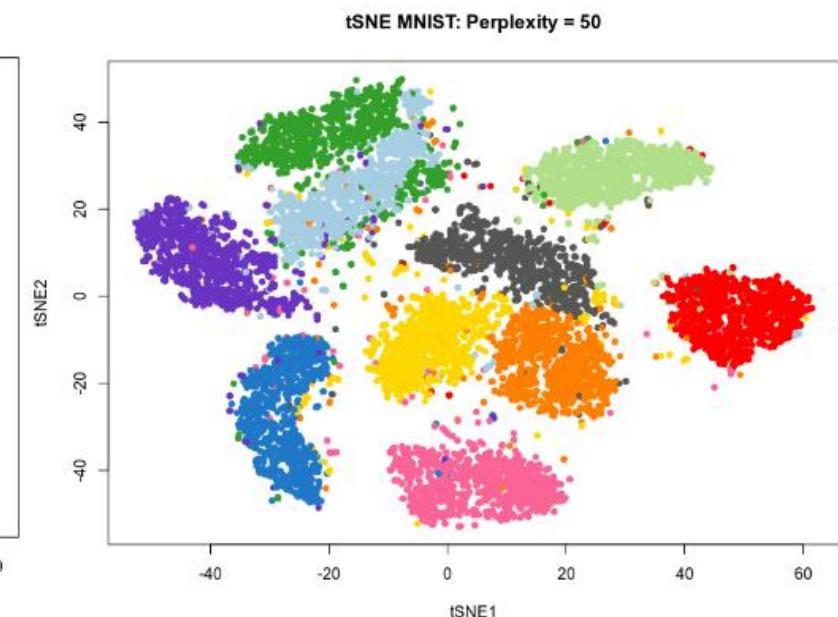
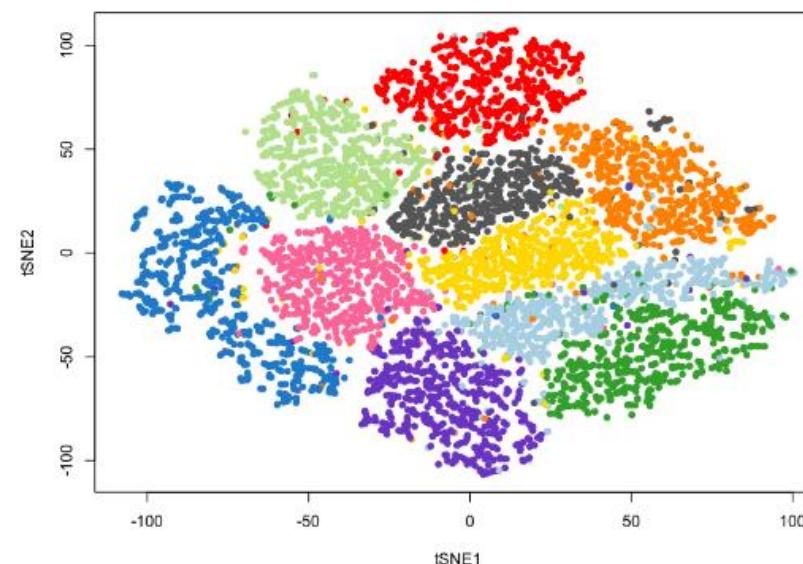


t-distributed stochastic neighbour embedding (t-SNE)

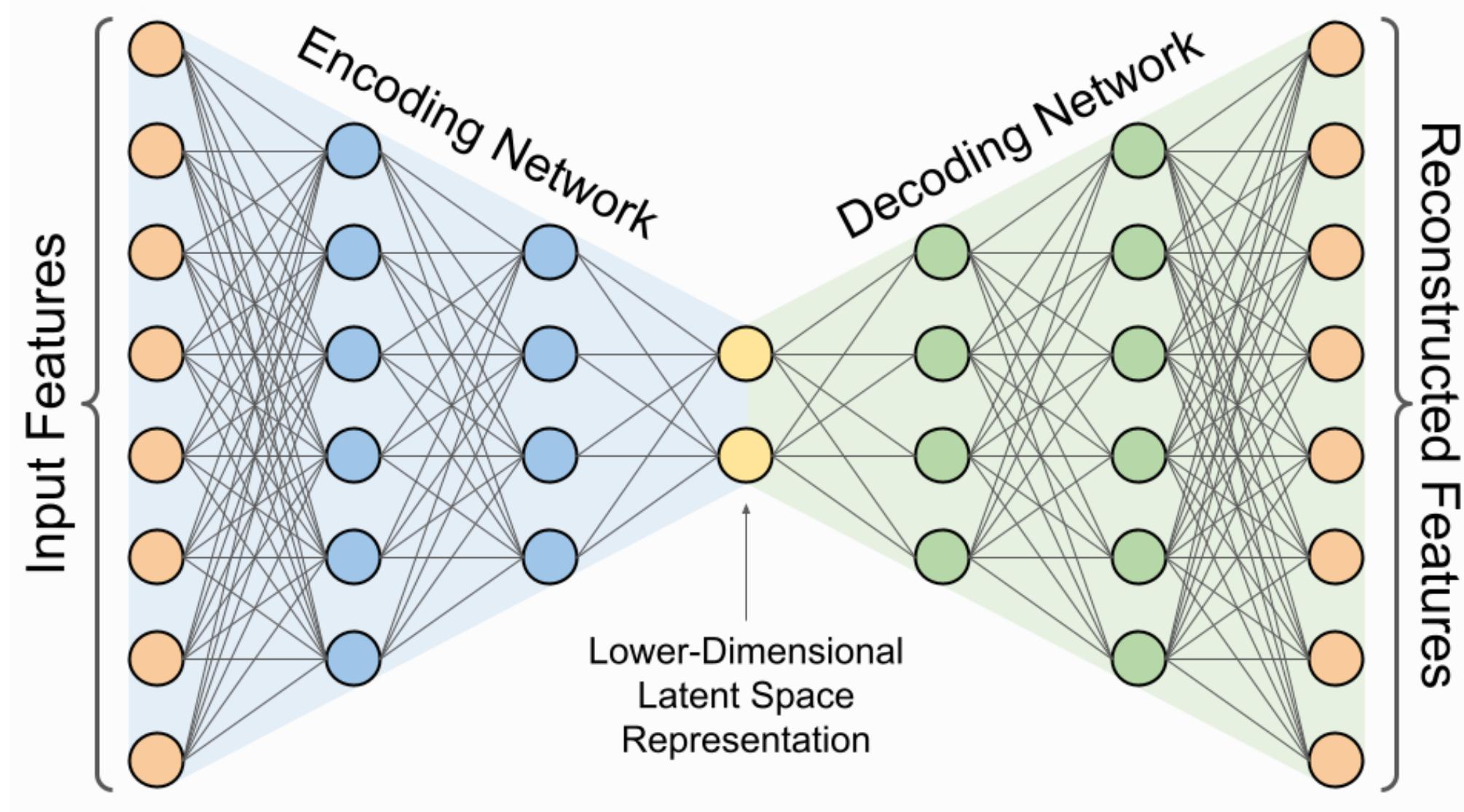
- Useful for *visualisation*, project high-dimensional data in 2 or 3 dimensions.
- Controlled by one main parameters: “perplexity”
- Relative distance between points not quantitatively meaningful

MNIST: database of written digits

0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9

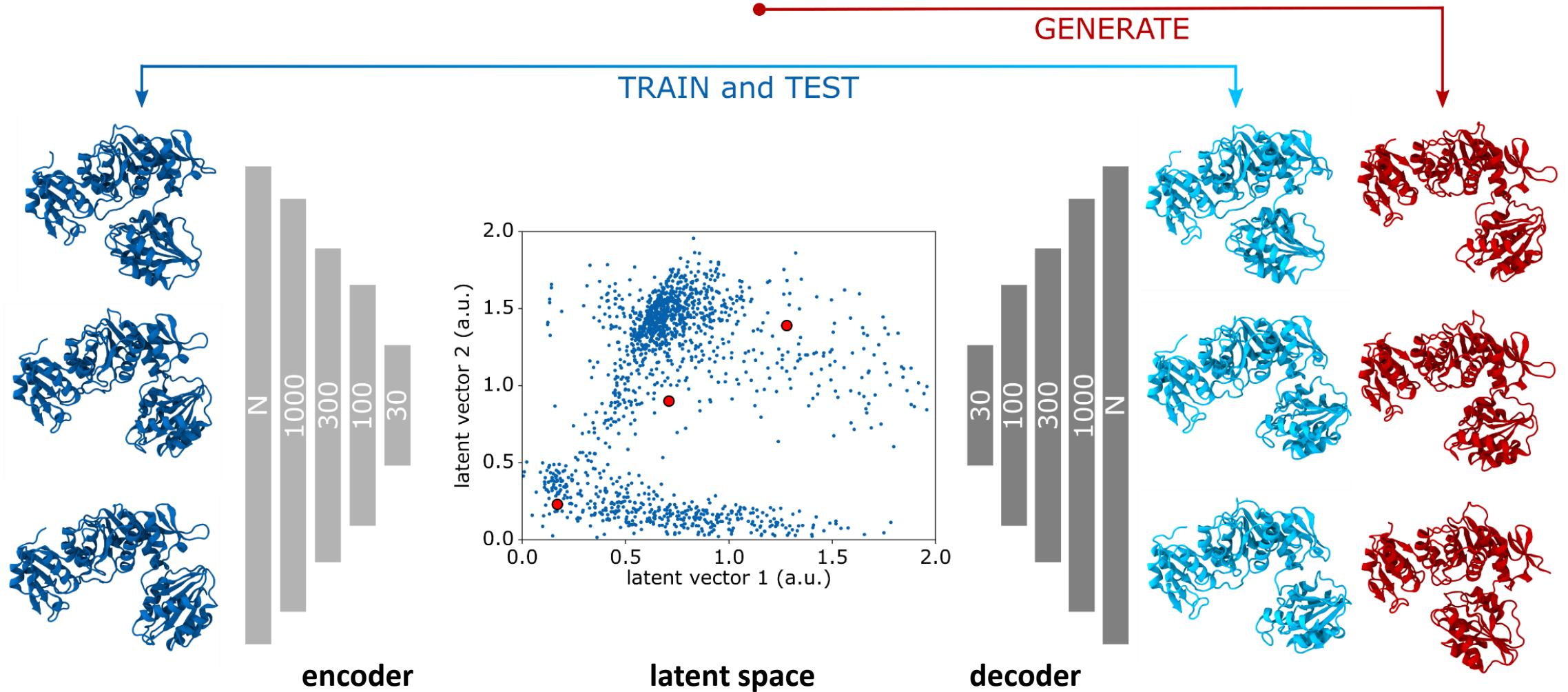


[Extra] Dimensionality reduction with neural networks



PCA dimensionality reduction is *interpretable*, that of t-SNE and neural networks is not. 12

[Extra] Learning Protein Conformational Space



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.

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

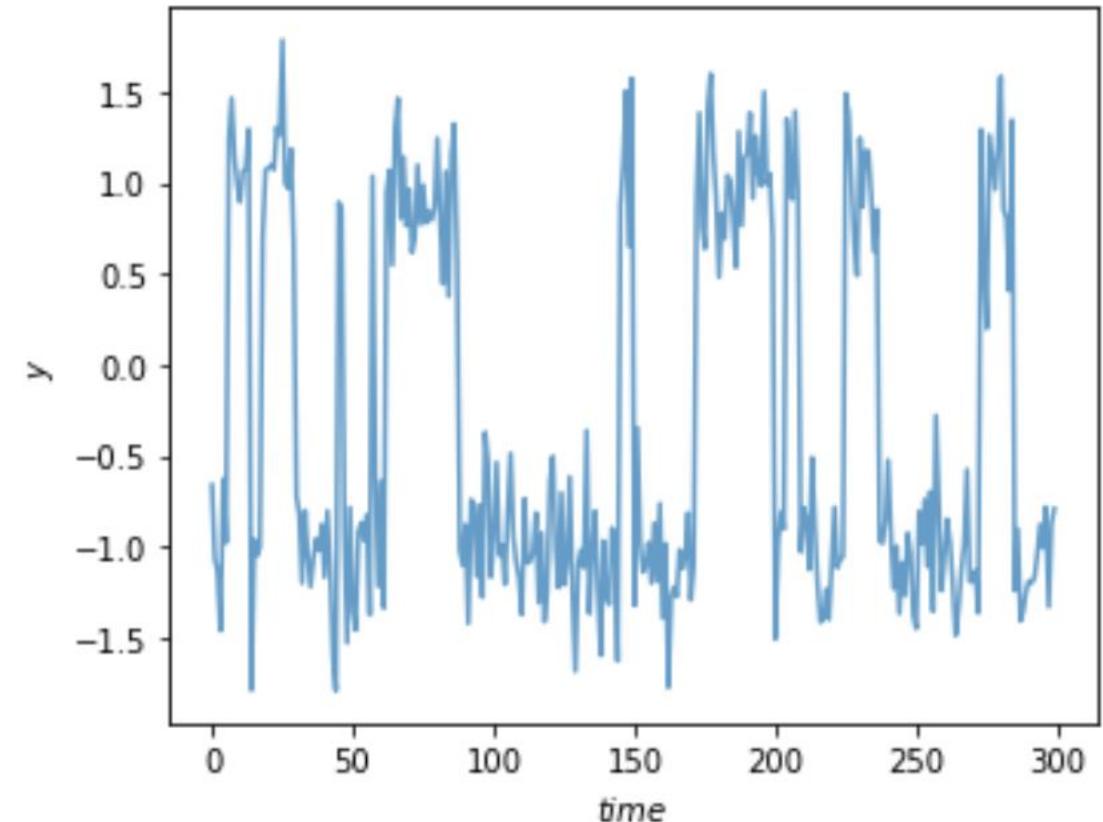
D-dimensional input data vector that is mean free, i.e., $\mathbf{r}(t) = \mathbf{r}(t) - \langle \mathbf{r}(t) \rangle_t$

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 (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:

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

We can now solve the generalised eigenvalue problem:

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

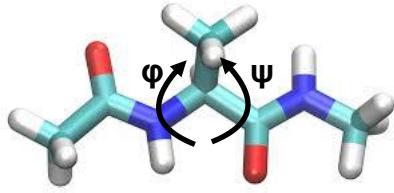
Eigenvector matrix containing ICs

Diagonal matrix with eigenvalues

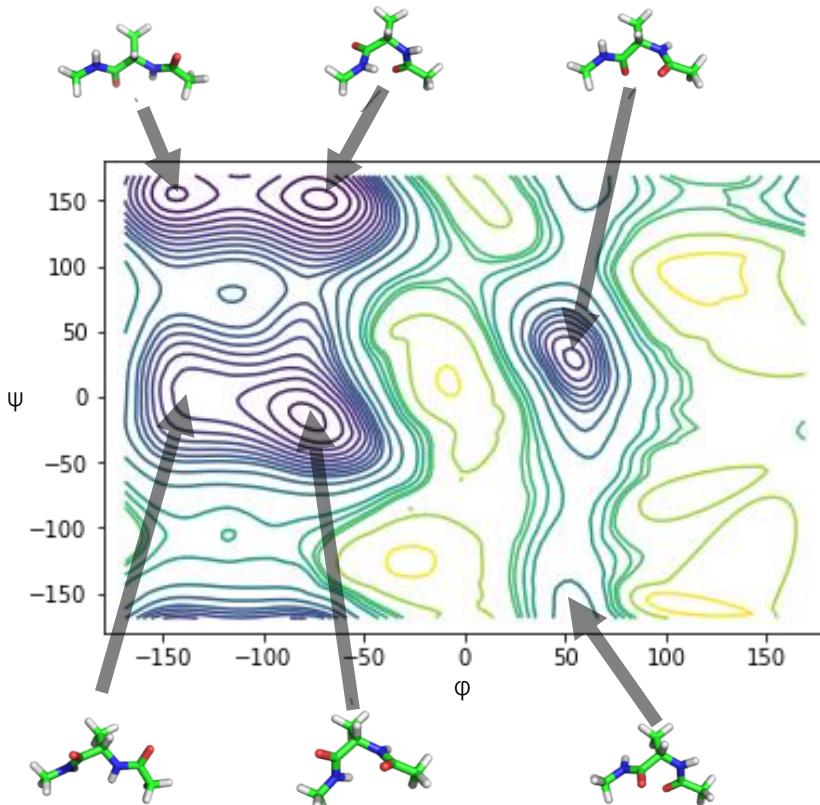
$$\mathbf{z}^\top(t) = \mathbf{r}^\top(t)\mathbf{U}$$

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

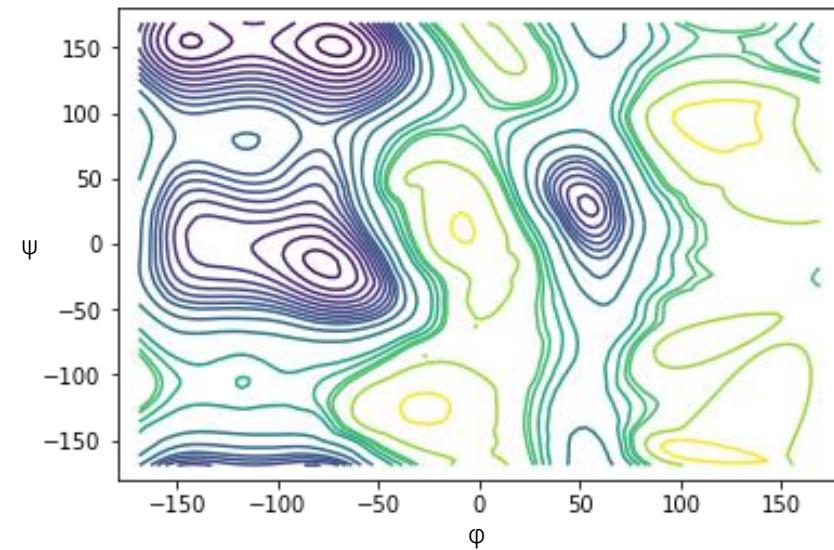
Dimensionality reduction for MD simulations



Alanine dipeptide



['PHI -150 ALA 2', 'PSI -150 ALA 2']
['PHI -150 ALA 2', 'PSI -50 ALA 2']
['PHI -50 ALA 2', 'PSI 50 ALA 2']
['PHI 50 ALA 2', 'PSI -150 ALA 2']
['PHI 50 ALA 2', 'PSI 0 ALA 2']
['PHI 0 ALA 2', 'PSI 0 ALA 2']



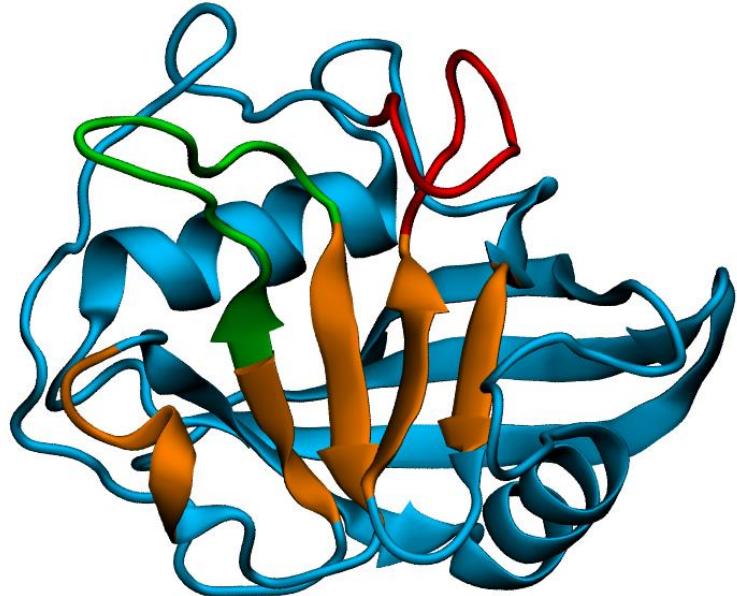
Dimensionality reduction for MD simulations



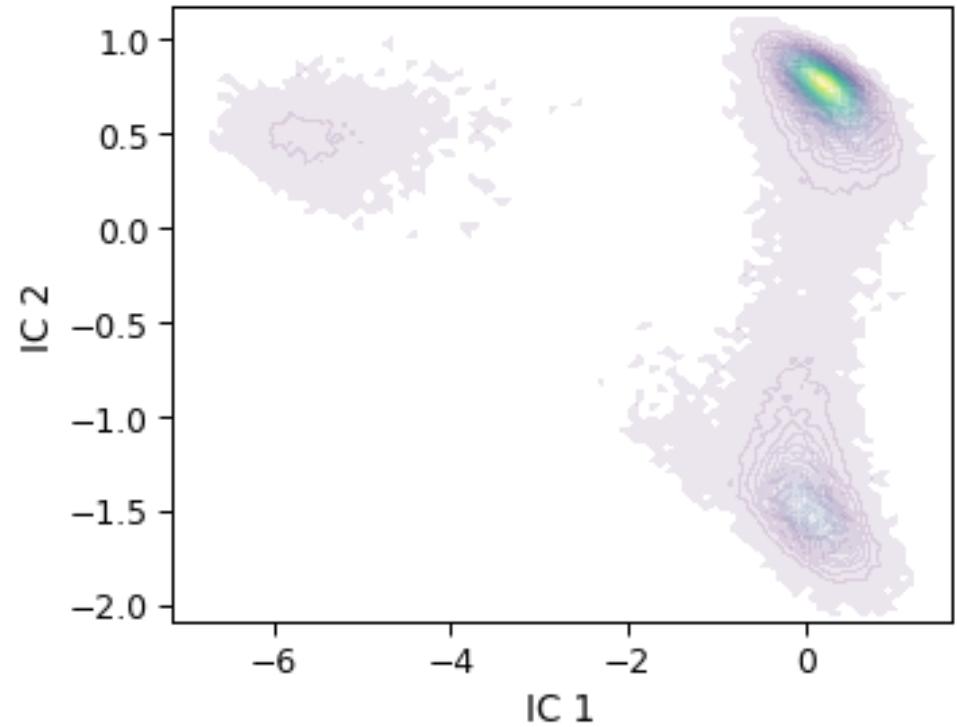
```
[ 'ATOM:ACE 1 CH3 1 x',
  'ATOM:ACE 1 CH3 1 y',
  'ATOM:ACE 1 CH3 1 z',
  'ATOM:ACE 1 C 4 x',
  'ATOM:ACE 1 C 4 y',
  'ATOM:ACE 1 C 4 z',
  'ATOM:ACE 1 O 5 x',
  'ATOM:ACE 1 O 5 y',
  'ATOM:ACE 1 O 5 z',
  'ATOM:ALA 2 N 6 x',
  'ATOM:ALA 2 N 6 y',
  'ATOM:ALA 2 N 6 z',
  'ATOM:ALA 2 CA 8 x',
  'ATOM:ALA 2 CA 8 y',
  'ATOM:ALA 2 CA 8 z',
  'ATOM:ALA 2 CB 10 x',
  'ATOM:ALA 2 CB 10 y',
  'ATOM:ALA 2 CB 10 z',
  'ATOM:ALA 2 C 14 x',
```

```
[ 'PHI 0 ALA 2', 'PSI 0 ALA 2' ]
```

Dimensionality reduction – TICA, PCA, VAMP



Project features onto
low dimensional
subspace

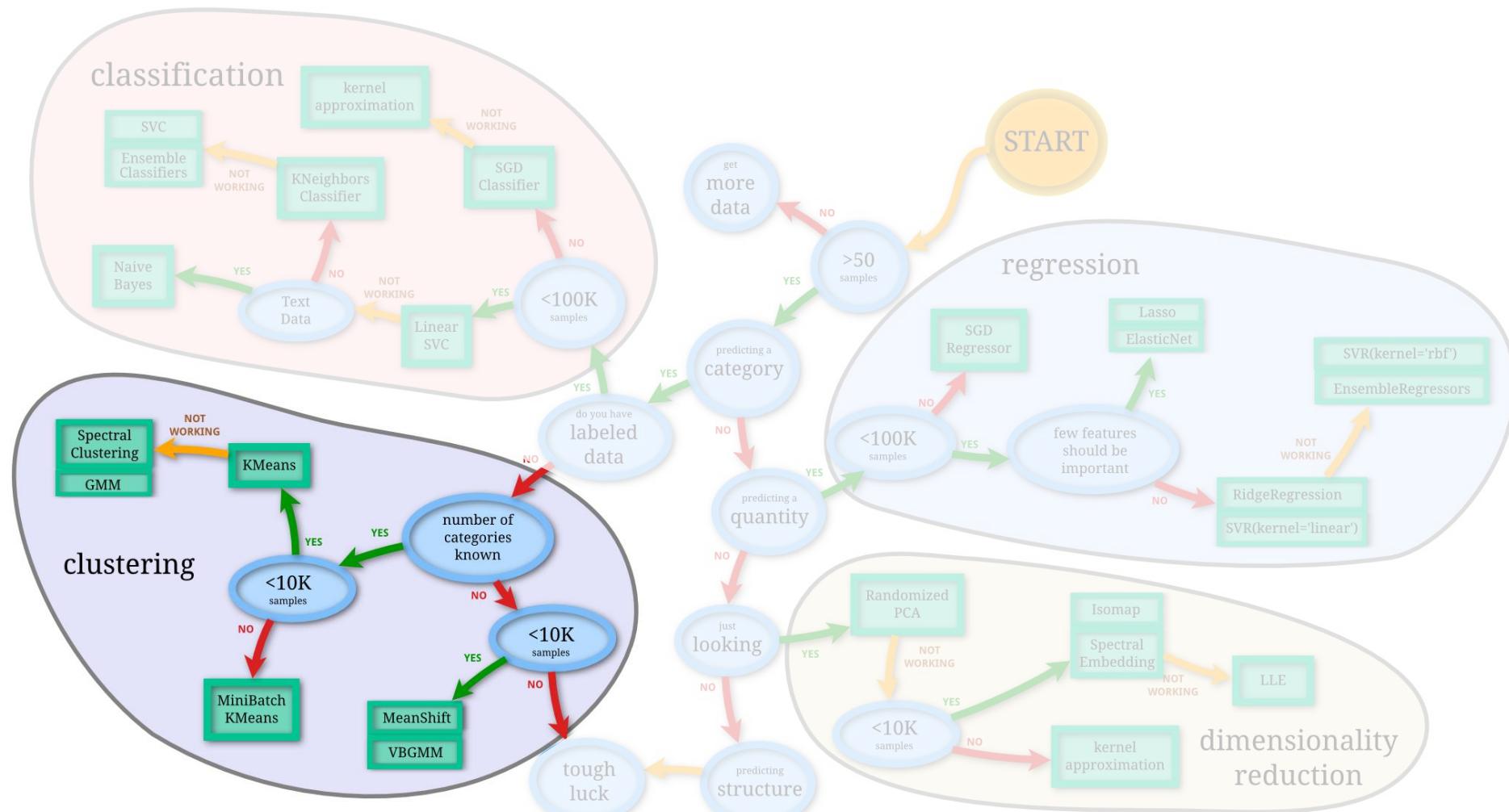


PCA: Linear combination of input features maximising the variance

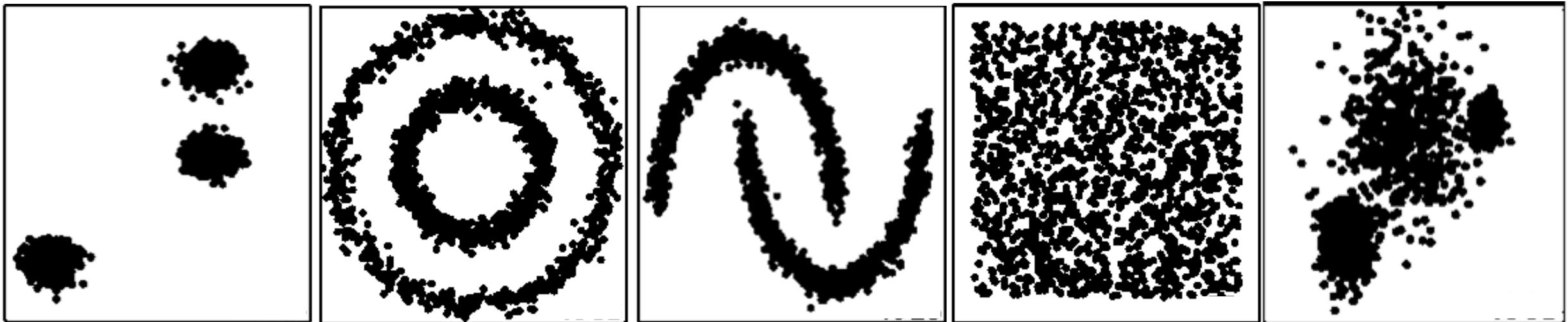
TICA: Linear combination of input features maximising time autocorrelation

VAMP: Variational approach for Markov Process, true for non-equilibrium data

The Data Mining world

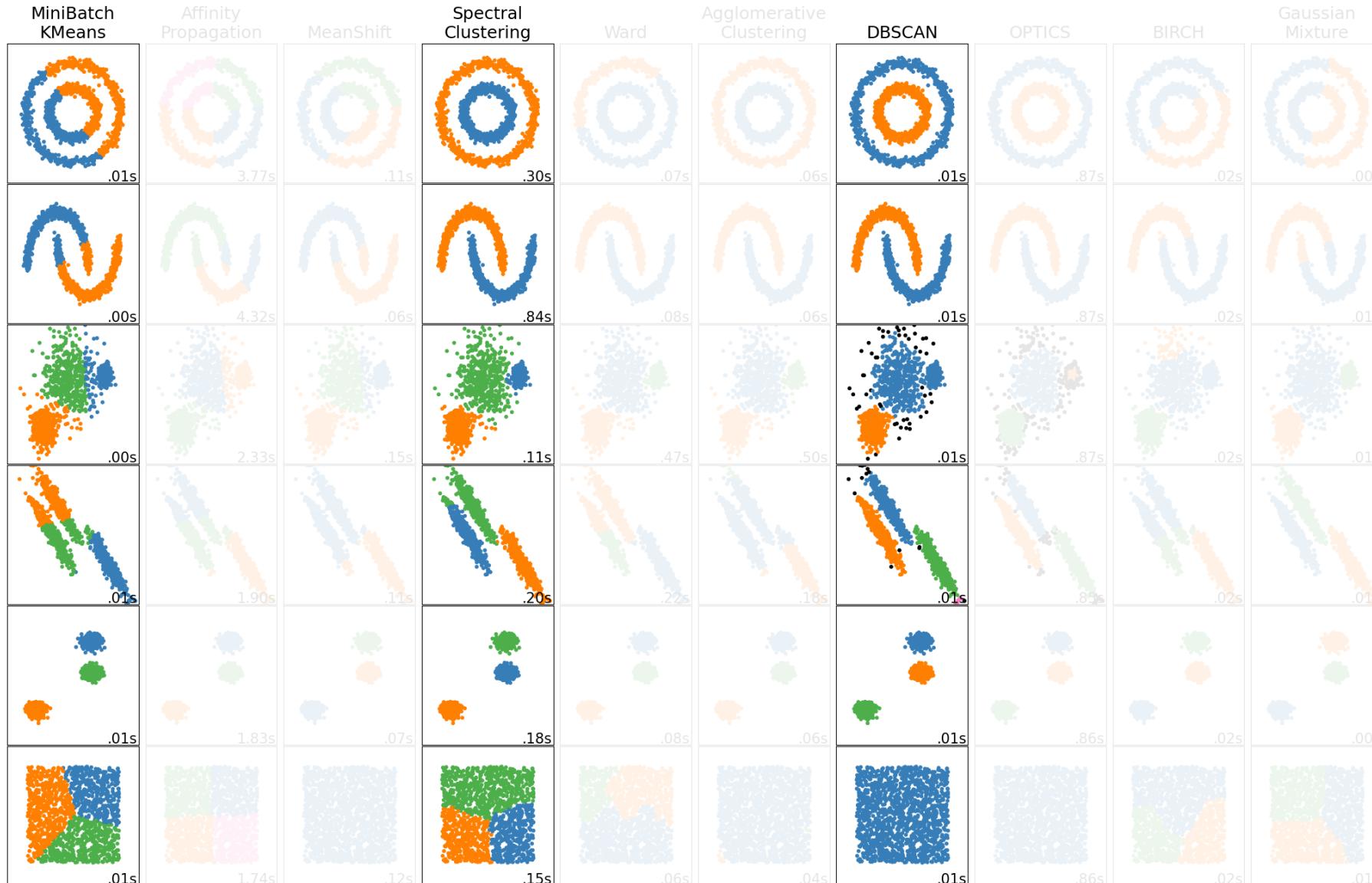


Clustering (i.e., unsupervised learning)



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

Clustering algorithms



How does k-means work?

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

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

Repeat until convergence:

For each point x_i :

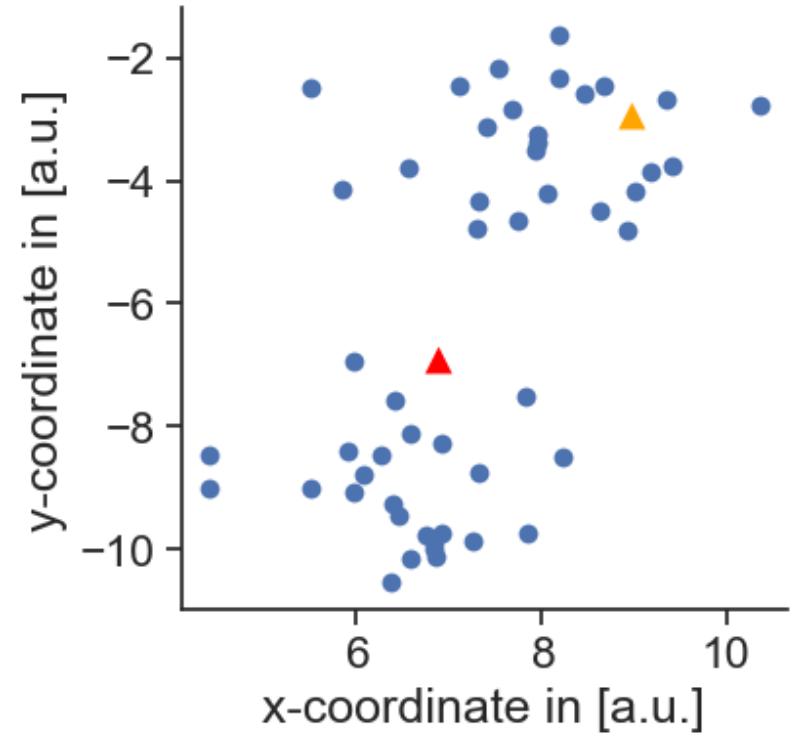
Find nearest centroid $c_j = \arg \min_j D(x_i, c_j)$

Assign the point x_i to cluster j

For each cluster $j = 1 \dots K$:

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

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



How does k-means work?

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

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

Repeat until convergence:

For each point x_i :

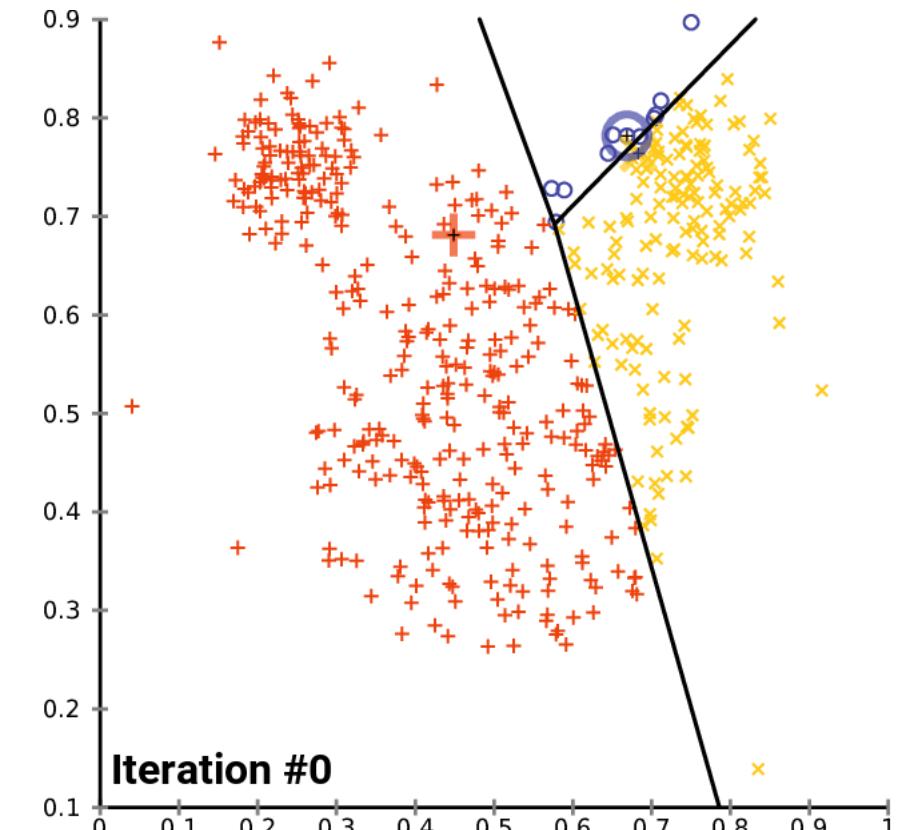
Find nearest centroid $c_j = \arg \min_j D(x_i, c_j)$

Assign the point x_i to cluster j

For each cluster $j = 1 \dots K$:

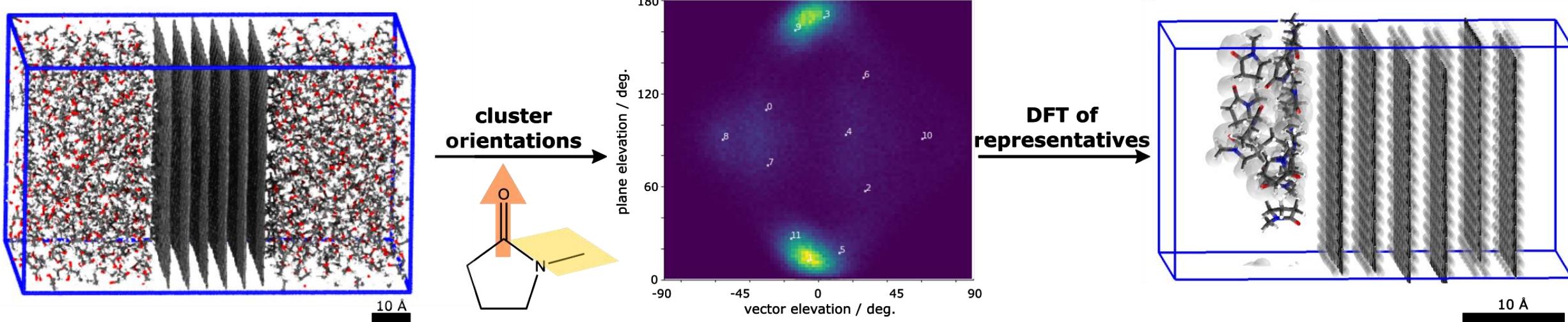
Compute the centroid mean for all points in one cluster and update the centroid

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



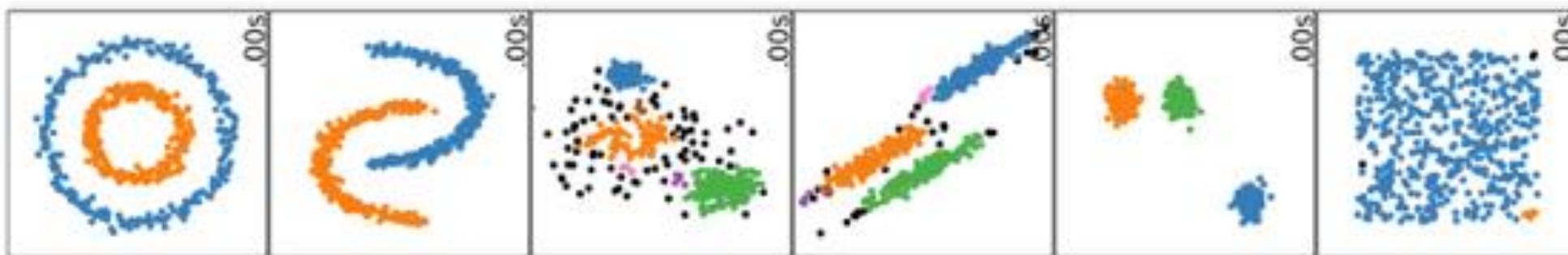
[Example] k-means vs solvent-graphite interactions

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

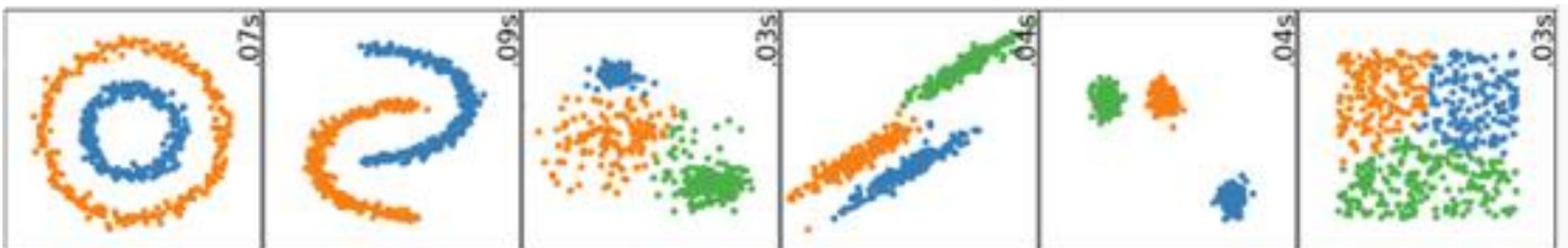


Density-based and spectral clustering

DBSCAN

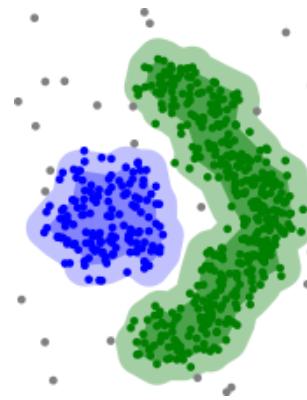


Spectral Clustering



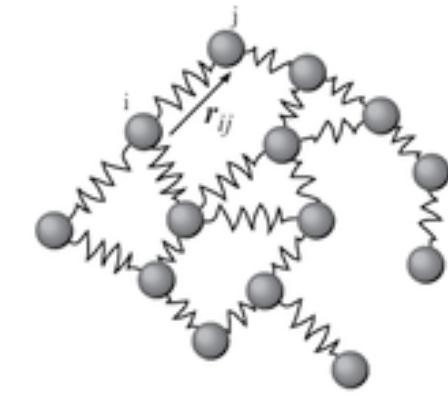
Density-based and spectral clustering

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 $\text{intensity} > t$
- Cluster beads and delete small clusters ($< 1\%$ of total beads)

