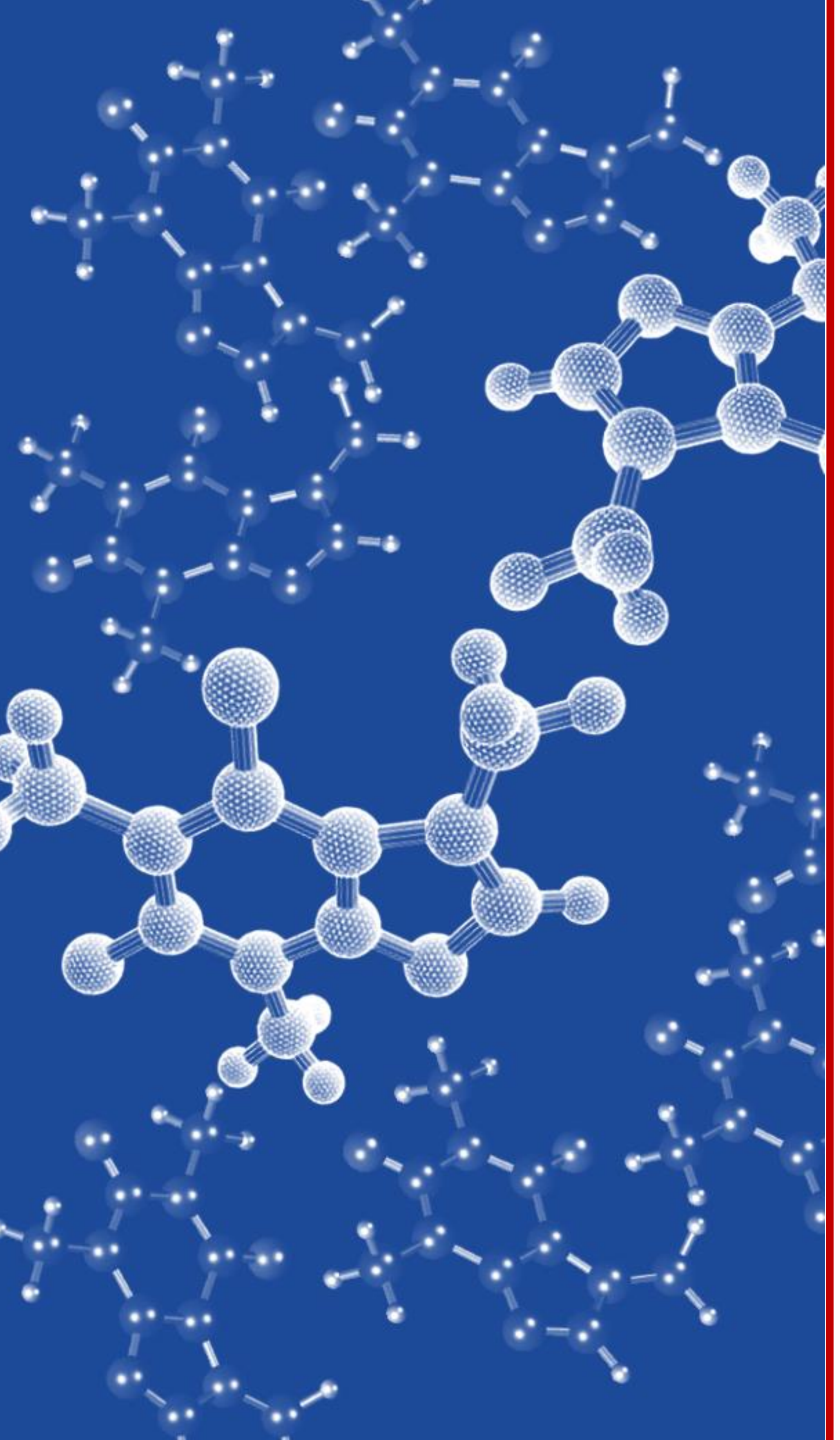


Unsupervised Learning for Chemistry: An introduction

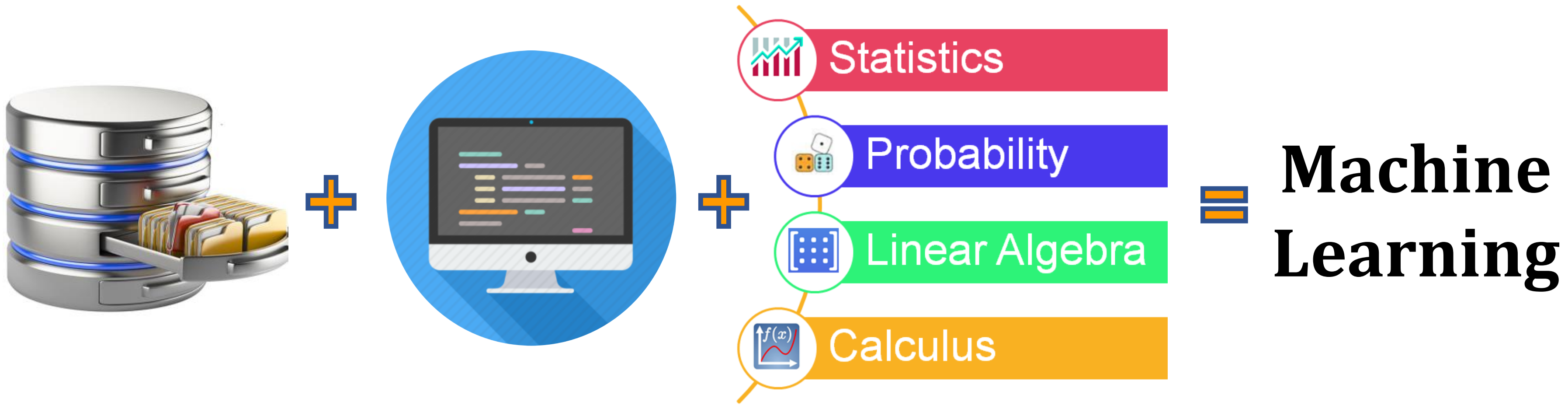
Max Pinheiro Junior

max.pinheiro-jr@univ-amu.fr

What is *Machine Learning?*

"The study of algorithms that allow computer programs to **automatically improve through experience** or exposure to data."

(Tom Mitchel, Machine Learning, McGraw Hill, 1997)

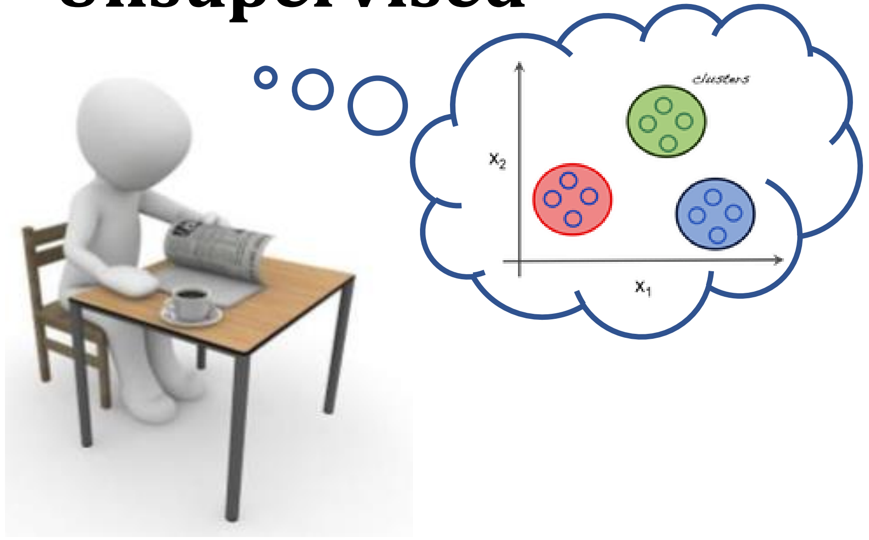


Machine Learning

Supervised

- **Knowledge of output** (labelled data)
- **Goal:** Predict a specific quantity (class or value)
- Can **measure accuracy** directly

Unsupervised



- **Unknown output** (unlabeled data)
- **Goal:** looking for structure or unusual patterns
- Indirect or qualitative evaluation

Learning chemistry from data

Data-driven approach

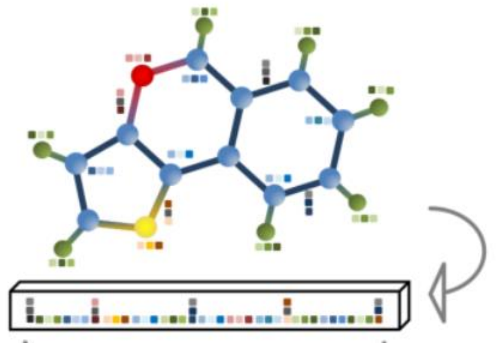
1. Convert molecules into **numeric inputs**

2. Data preprocessing

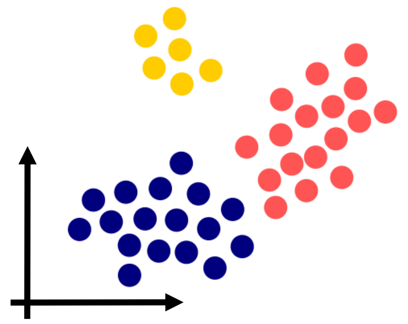
3. Train or choose a **ML model**

4. **Evaluate** predictions or model outcomes

Representation

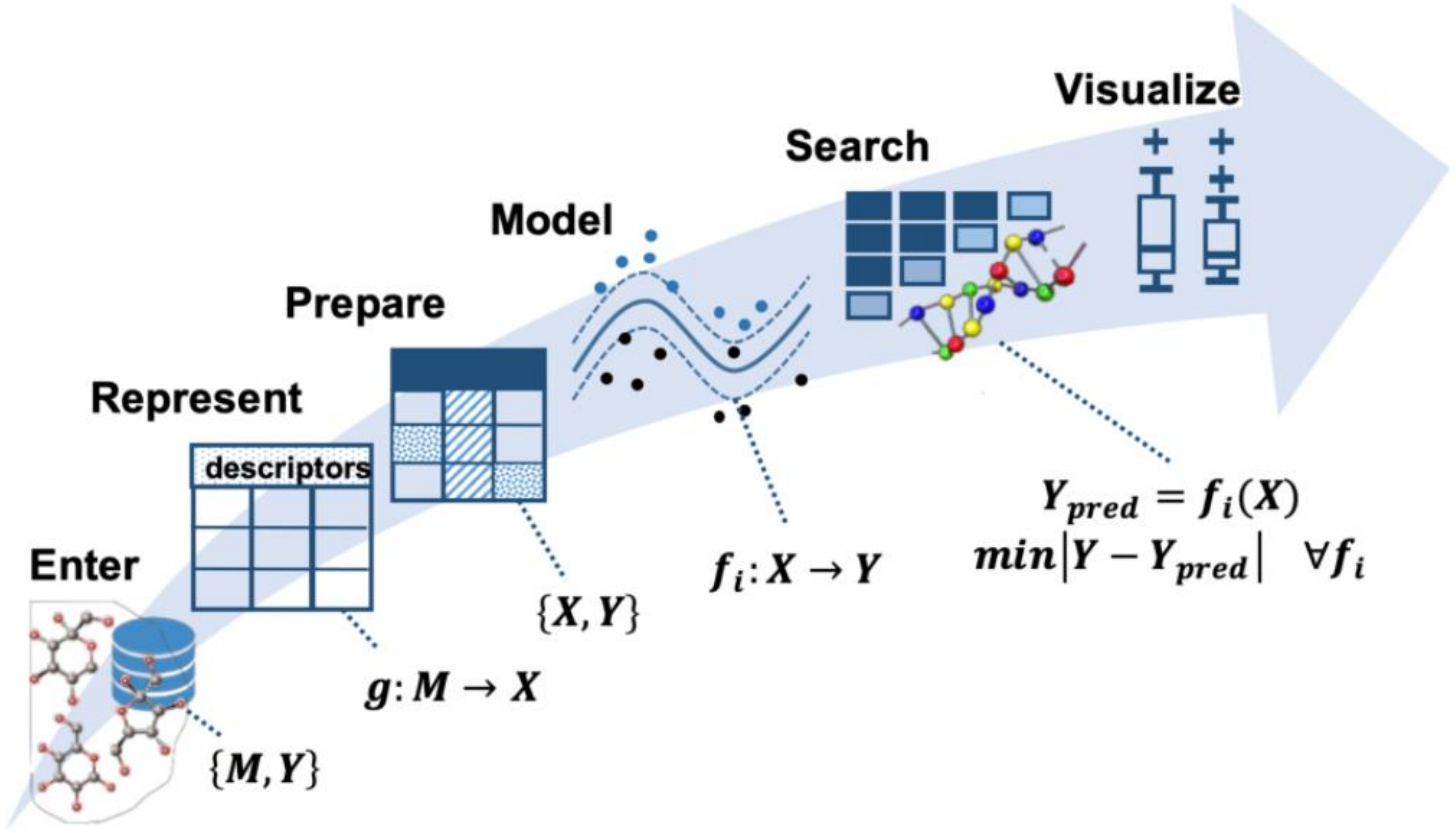


Learning Model

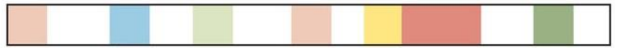


Metrics for evaluation

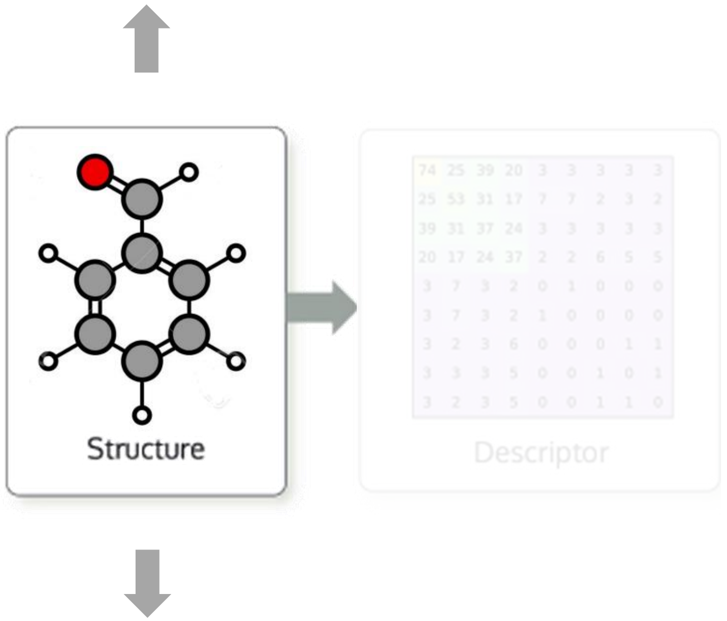
Machine Learning Pipeline



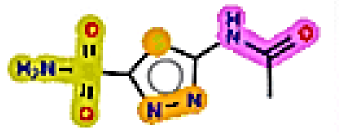
Molecular representations



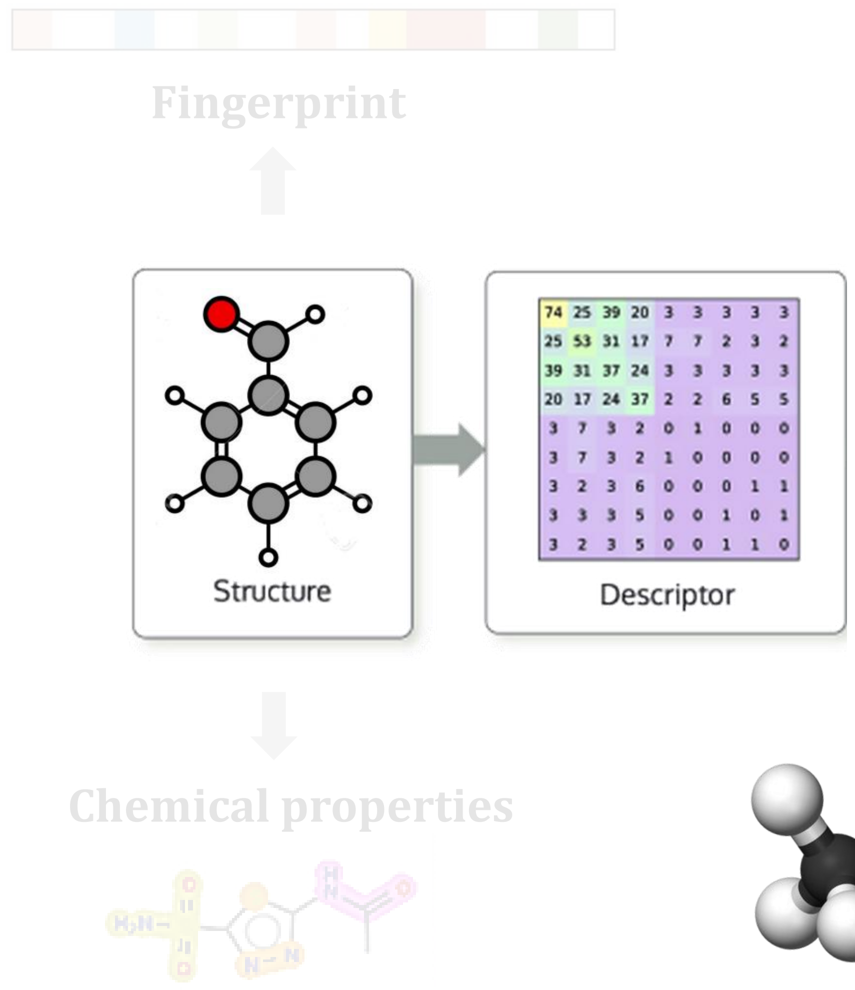
Fingerprint



Chemical properties



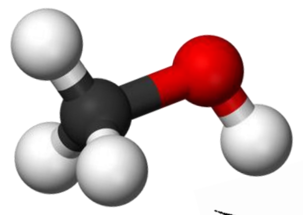
Molecular representations



Coulomb matrix

Simulates electrostatic interactions between atoms

$$M_{ij}^{\text{Coulomb}} = \begin{cases} 0.5 Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{for } i \neq j \end{cases}$$



$$\begin{array}{c|cccccc} \text{C} & 36.9 & 33.7 & 5.5 & 3.1 & 5.5 & 5.5 \\ \text{O} & 33.7 & 73.5 & 4.0 & 8.2 & 3.8 & 3.8 \\ \text{H} & 5.5 & 4.0 & 0.5 & 0.35 & 0.56 & 0.56 \\ \text{H} & 3.1 & 8.2 & 0.35 & 0.5 & 0.43 & 0.43 \\ \text{H} & 5.5 & 3.8 & 0.56 & 0.43 & 0.5 & 0.56 \\ \text{H} & 5.5 & 3.8 & 0.56 & 0.43 & 0.56 & 0.5 \end{array}$$

M. Rupp *et al*, ***Phys. Rev. Lett.*** (2012) 108, 058301

Toolbox for Machine Learning

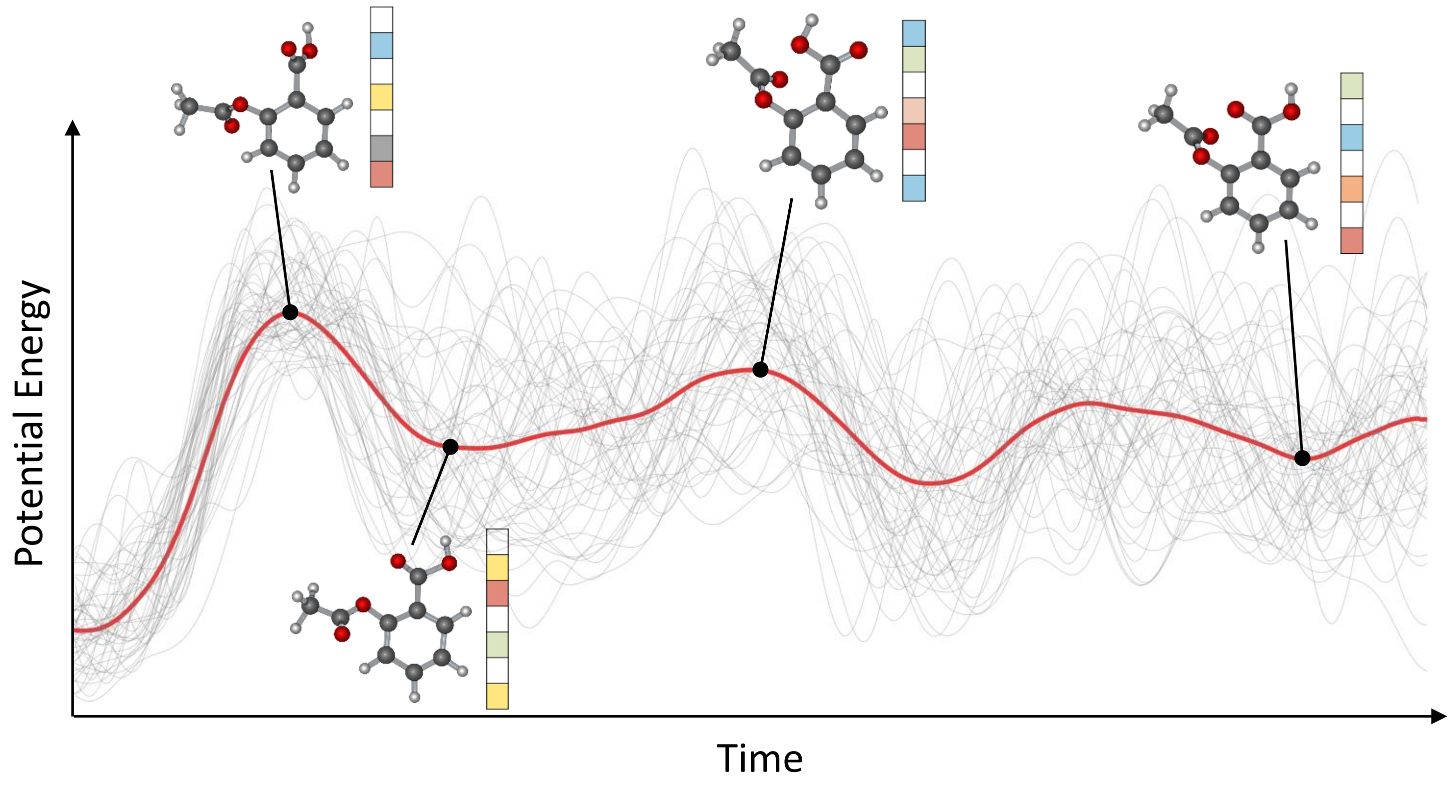
General **Python** Libraries

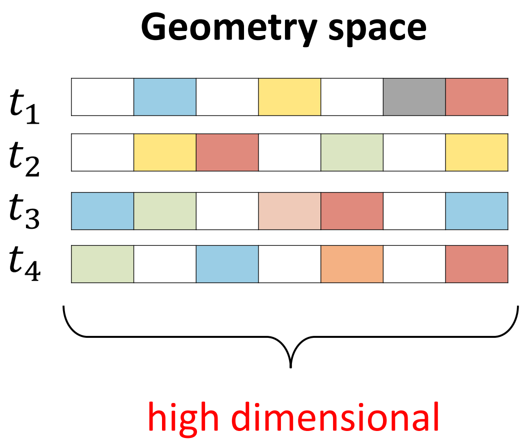
Data	pandas
Vis.	matplotlib, seaborn
Math.	NumPy, SciPy
Modeling	scikit learn

For Chemistry

DScribe
TORCHANI
deepchem
Schnetpack, QML, ...
MLatom.com
RDKit
Open-Source Cheminformatics and Machine Learning

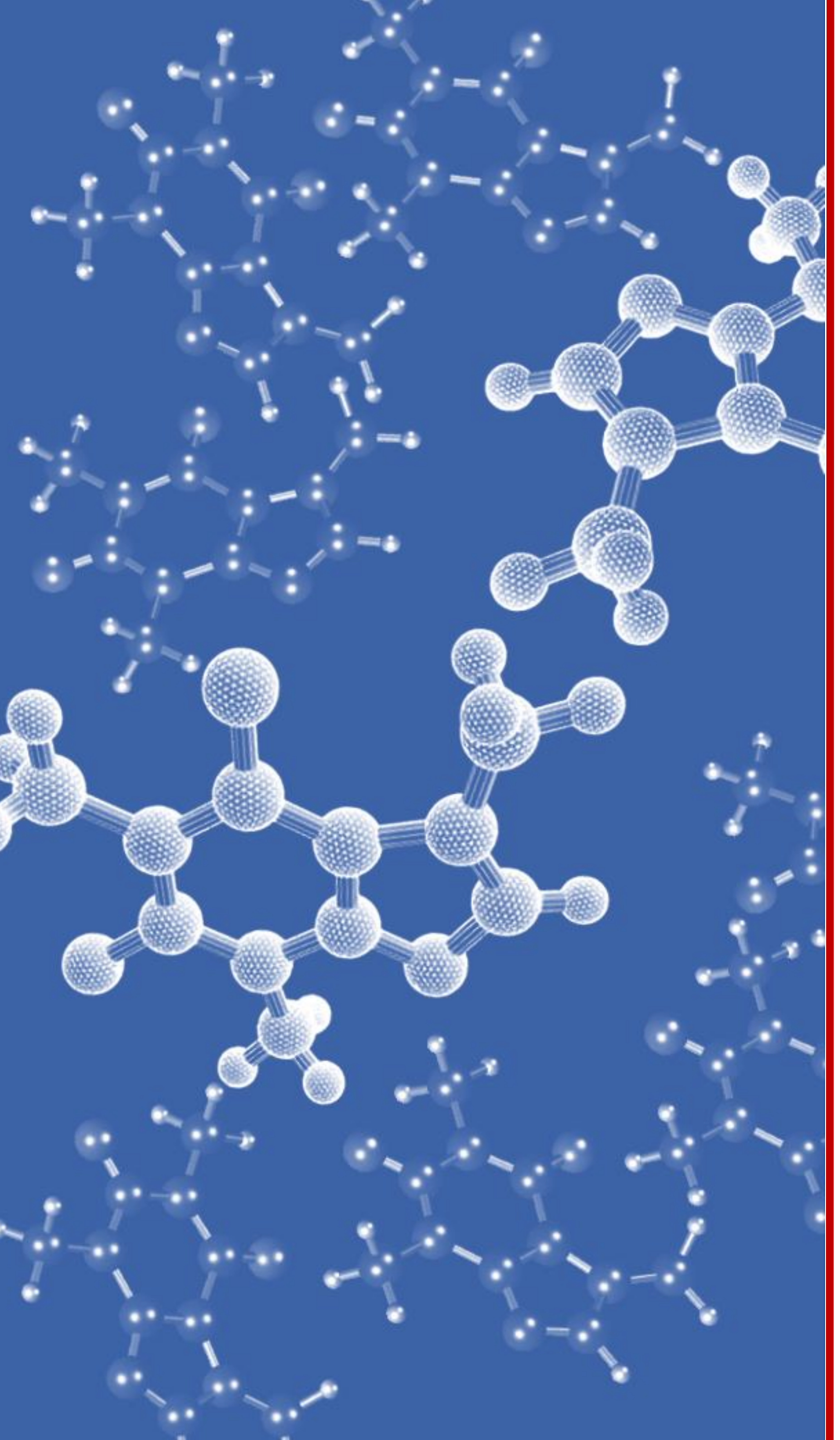
Analyzing molecular dynamics data





How to visualize?

How to find similar geometries?

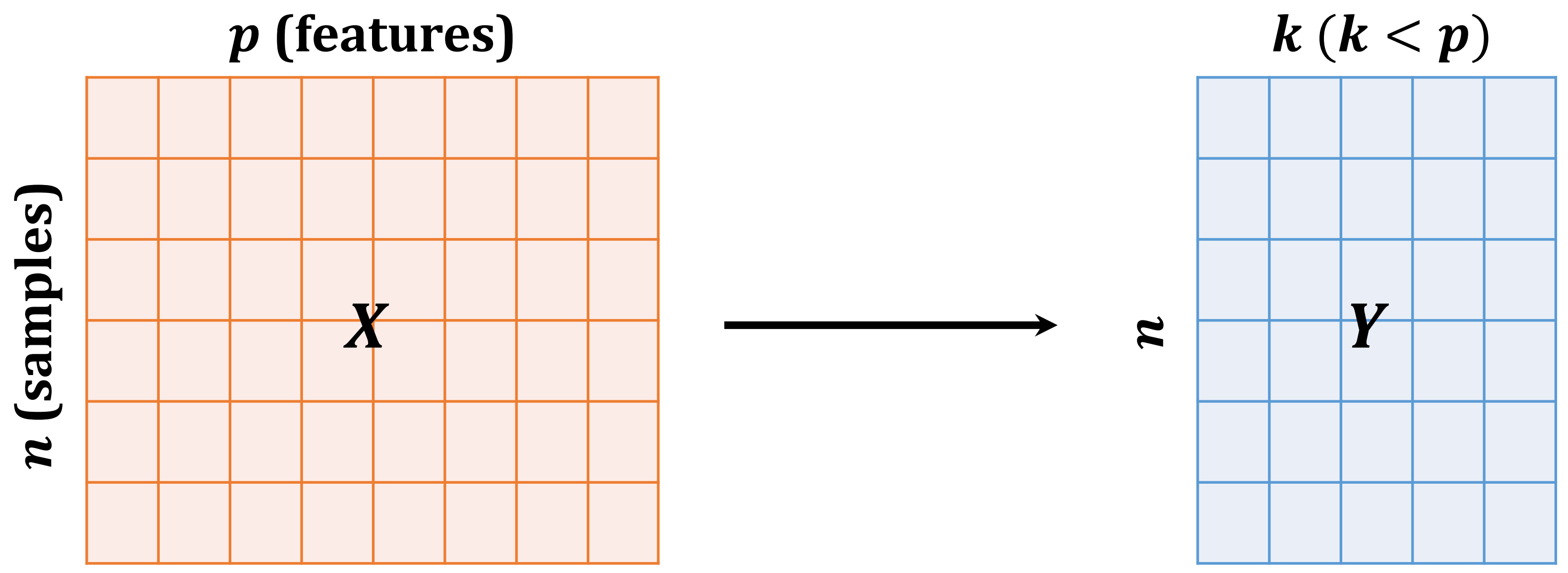


Part 1:
Dimensionality Reduction

Why reduce dimensions?

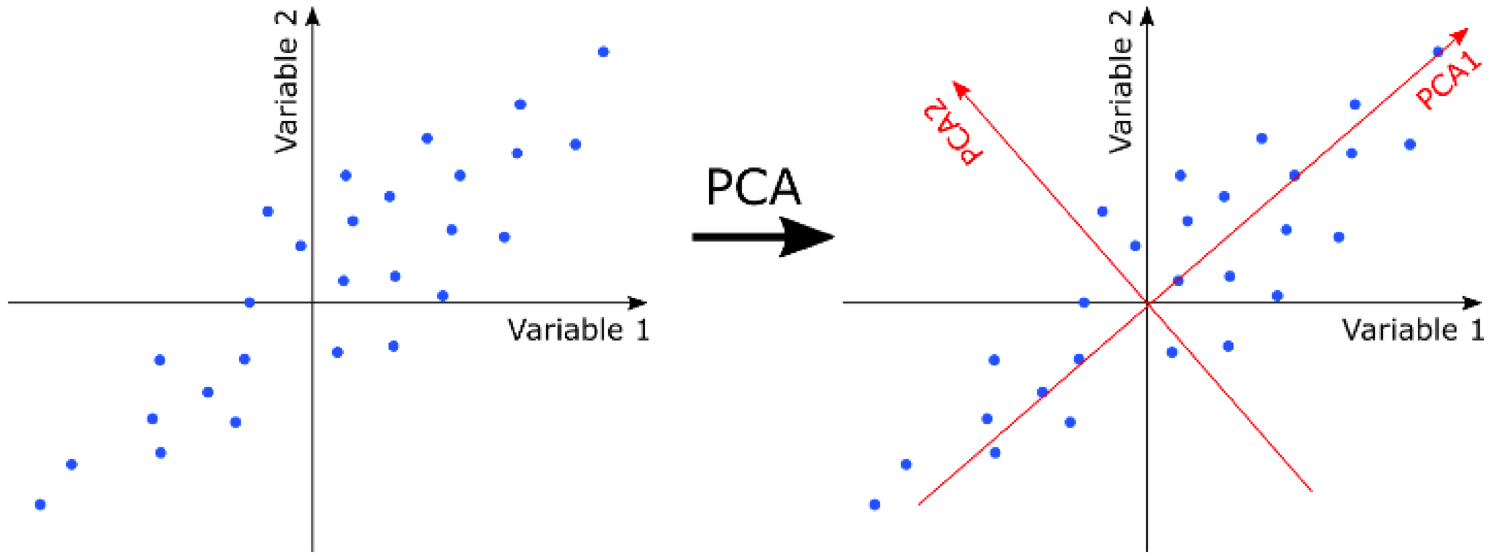
“… dimensionality reduction yields a more compact, more easily interpretable **representation of the data**, focusing the attention on the **most relevant variables**.”

— Page 289, Data Mining: Practical Machine Learning Tools and Techniques, 4th edition, 2016.



Principal Component Analysis

Main goal: decompose a dataset into a set of **orthogonal components** that explains a **maximum** amount of **variance**.



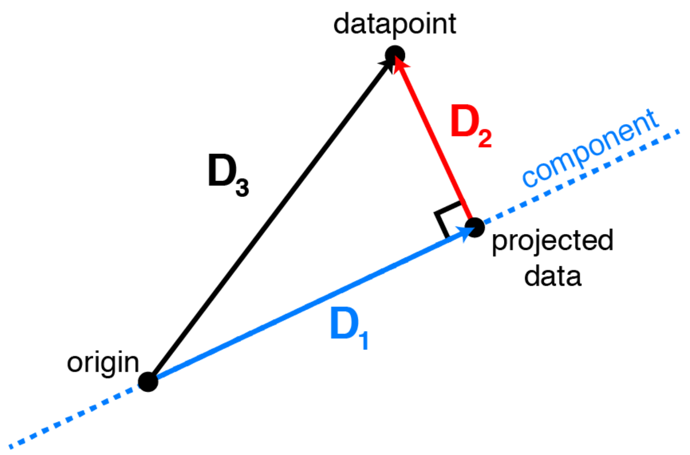
https://setosa.io/ev/principal-component-analysis/

PCA in a nutshell

PC as a new basis: $\mathbf{c}_{i1} = w_{11}x_{i1} + \ldots + w_{p1}x_{ip}$

Constrain (PCs normalized): $\sum_{j=1}^{p} w_{j1}^2 = 1$

Objective: Find w's that **maximize the variance** $\max_{w_{11},\ldots,w_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathbf{c}_{i1}^2 \right\}$ **or** **minimize the reconstruction error** $\min_{\mathbf{W}} \| \mathbf{X} - \mathbf{X}\mathbf{C}\mathbf{C}^{\mathbf{T}} \|_F^2$ (information loss).



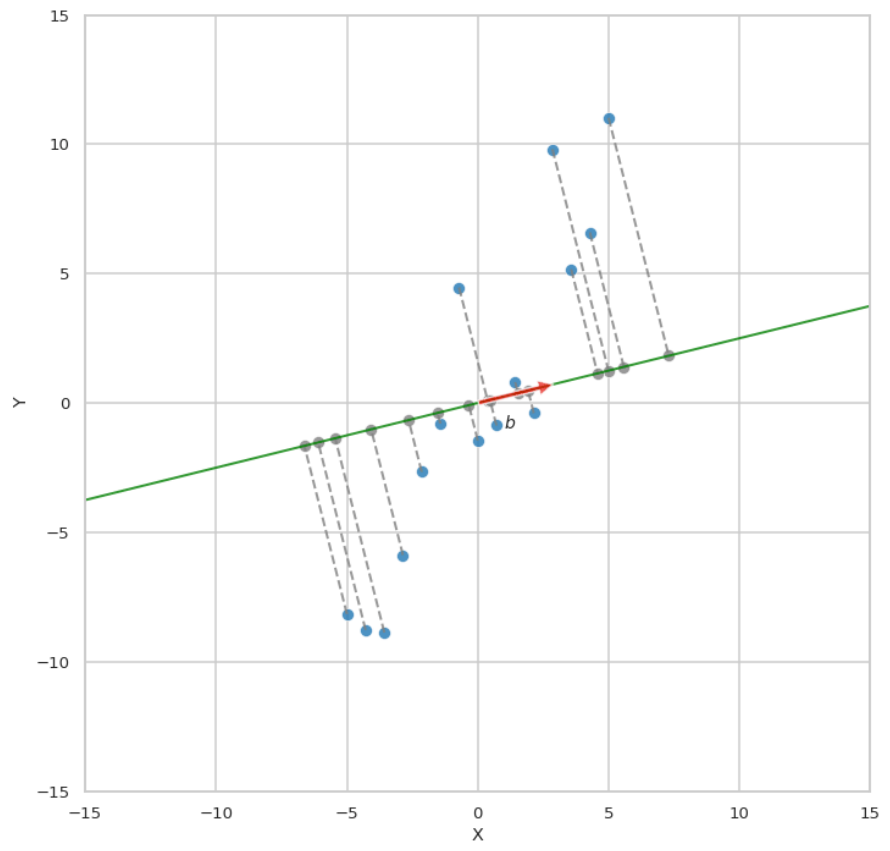
$$D_3^2 = D_1^2 + D_2^2$$

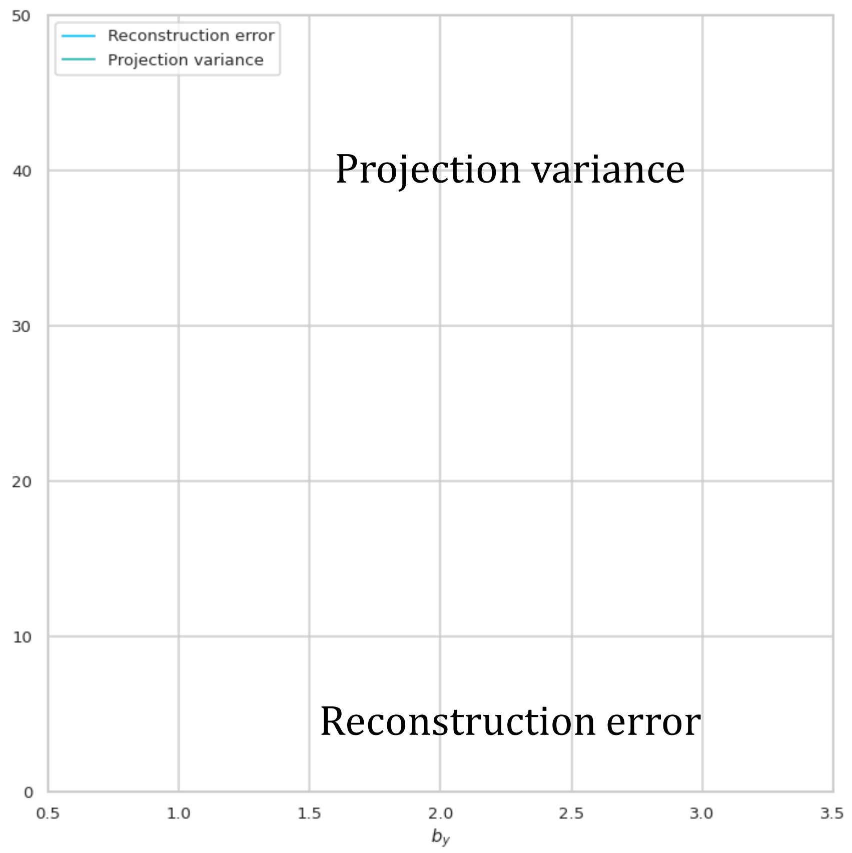
initial variance = **remaining variance** + **lost variance**

$$\|\mathbf{a}_i\|^2 = \|w_i\,\mathbf{c}\|^2 + \|\mathbf{a}_i - w_i\,\mathbf{c}\|^2$$

this is constant | maximize this **or** minimize this

PCA in a nutshell





https://towardsdatascience.com/dimensionality-reduction-with-pca-from-basic-ideas-to-full-derivation-37921e13cae7

PCA in a nutshell

PC as a new basis: $\mathbf{c}_{i1} = w_{11}x_{i1} + \ldots + w_{p1}x_{ip}$

Constrain (PCs normalized): $\sum_{j=1}^{p} w_{j1}^2 = 1$

Objective: Find w's that **maximize the variance** $\max_{w_{11},\ldots,w_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \mathbf{c}_{i1}^2 \right\}$ **or** **minimize the reconstruction error** $\min_{\mathbf{W}} \left\| \mathbf{X} - \mathbf{X}\mathbf{C}\mathbf{C}^{\mathbf{T}} \right\|_F^2$ (information loss).

Solution

Compute Eigenvalues and Eigenvectors of covariance matrix Σ:

$\Sigma c_i = \lambda_i c_i \; (i = 1, 2, 3, \ldots, p),$ $\Sigma = \frac{1}{n} \sum_{k=1}^{n} x_k^T x_k$: Covariance matrix

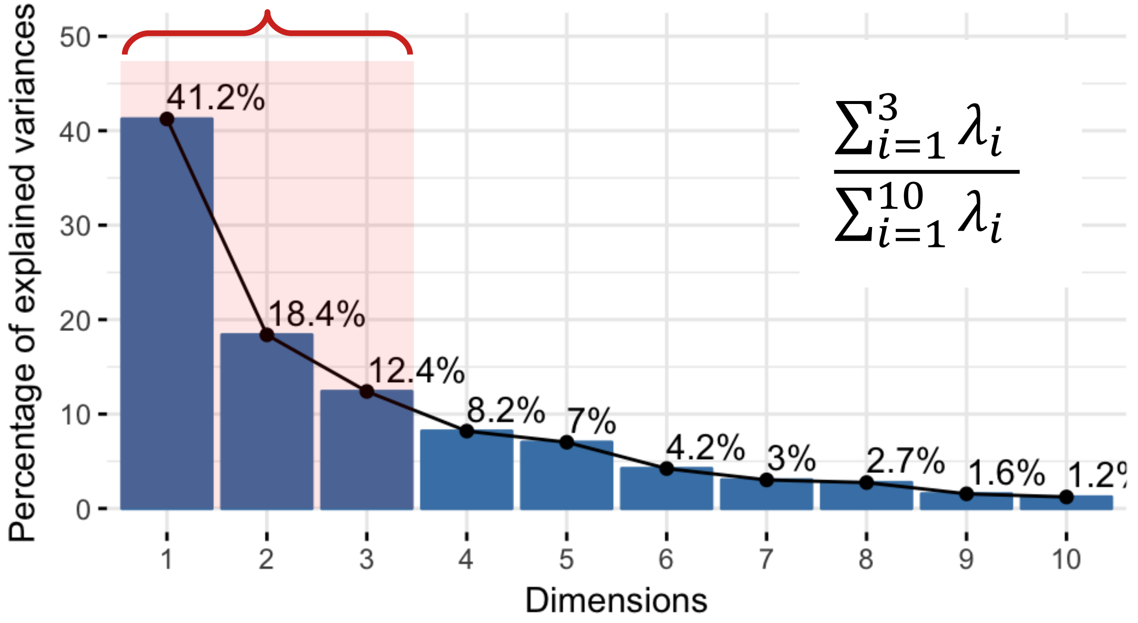
$\xrightarrow{}$ Principal Component

PCA: explained variance

How to select the optimal number of components?

- Total variance = sum of variances of all individual principal components.
- The fraction of variance explained by a PC is the ratio between the variance of that PC and the total variance

$PC1 + PC2 + PC3 = 72\%$ **of total variance**



$$\frac{\sum_{i=1}^{3}\lambda_i}{\sum_{i=1}^{10}\lambda_i}$$

PCA Algorithm

1. Compute the mean feature vector: $\mu = \frac{1}{p}\sum_{k=1}^{p} x_k$

2. Compute the covariance matrix: $\Sigma = \frac{1}{p}\sum_{k=1}^{p} (x_k - \mu)^T (x_k - \mu)$

3. Compute Eigenvalues and Eigenvectors of Σ:

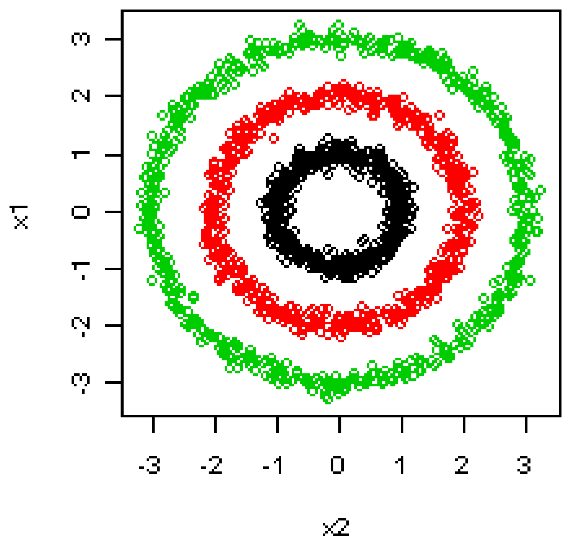
$$\Sigma c_i = \lambda_i c_i \ (i = 1, 2, 3, \dots, p), \qquad p = \text{number of features}$$

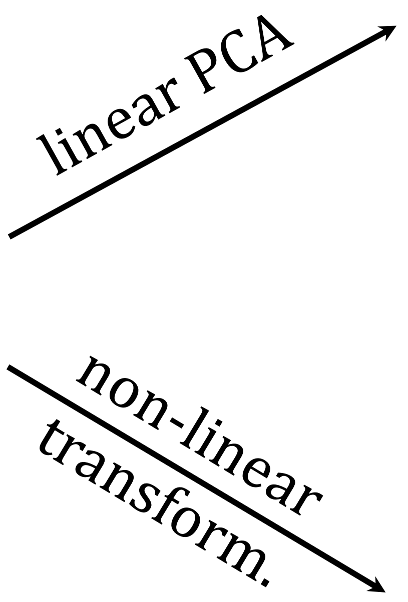
4. Select the k eigenvectors c_i corresponding to the largest eigenvalues λ_i

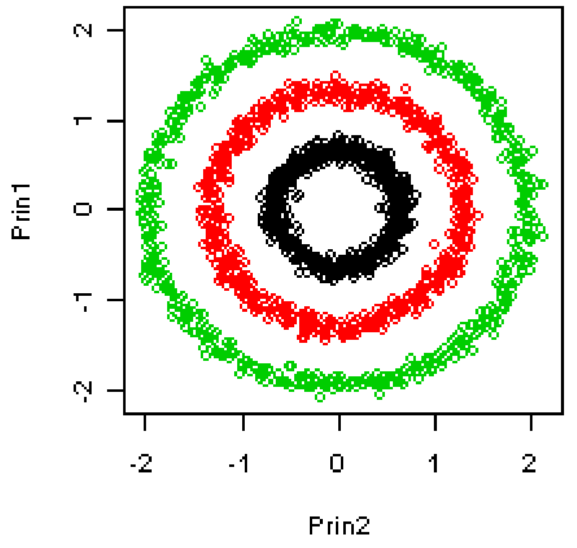
5. Reduce dimension from p to k with $Y = XC,\ C = (c_1, \dots, c_k)$

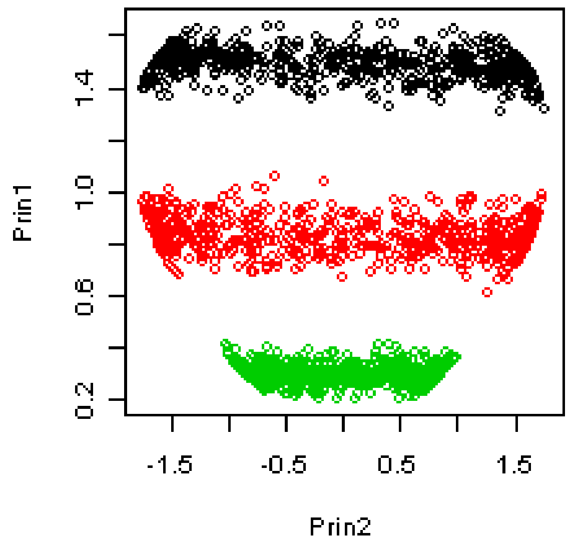
PCA on non-linear data?

Data with isotropic variance







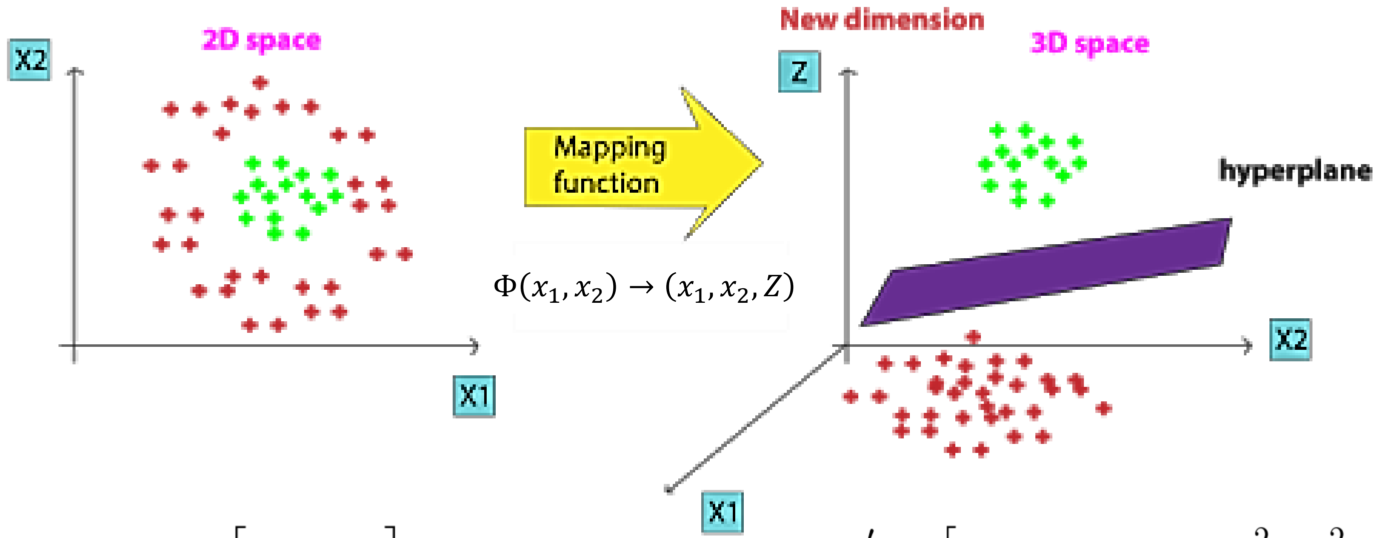


Kernel Principal Component Analysis

PCA on non-linear data?

Main idea: map data onto a **high-dimensional feature space** (non-linear combinations) where data can be **linearly separable**

Apply a non-linear transformation on the data: $\mathbf{x} \rightarrow \Phi(\mathbf{x})$



$\mathbf{x} = [x_1 \quad x_2]$

$\mathbf{x}' = [x_1 \quad x_2 \quad x_1 x_2 \quad x_1^2 \quad x_2^2 \quad \ldots]$

Kernelized PCA

Trick: No need to know the mapping Φ(x) explicitly!

Instead: Use a **kernel (similarity) matrix** that can be expressed as dot products in the (HD) feature space.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \Phi(\boldsymbol{x}_i), \Phi(\boldsymbol{x}_j) \rangle = \begin{pmatrix} \Phi(x_1)\Phi(x_1) & \cdots & \Phi(x_1)\Phi(x_N) \\ \Phi(x_2)\Phi(x_1) & \cdots & \Phi(x_2)\Phi(x_N) \\ \vdots & \ddots & \vdots \\ \Phi(x_N)\Phi(x_1) & \cdots & \Phi(x_N)\Phi(x_N) \end{pmatrix}$$

Examples of kernels:

Kernel function	Expression	Parameter
Polynomial kernel function	$K(x_i, x_j) = (x_i \cdot x_j + 1)^d$	d
Radial basis function (RBF) kernel function	$K(x_i, x_j) = \exp\left(-\gamma \lVert x_i - x_j \rVert^2\right)$	$\gamma > 0$

KPCA algorithm

1. Choose a kernel function – $K(x, x')$
   (ex: gaussian, polynomial, sigmoid, cosine)

2. Compute the kernel (similarity) matrix K for all data points

3. Centralize the kernel matrix: $\widetilde{K} = K - \underbrace{\mathbf{1_N}}_{\text{Matrix with entries } 1/N} K - K\mathbf{1_N} + \mathbf{1_N}K\mathbf{1_N}$

4. Solve the eigenvalue equation: $\widetilde{K}\alpha_i = \lambda_i \alpha_i$

5. Select the n largest eigenvalues to obtain the PCs in the feature space

$$y_j = \sum_{i=1}^{d} \alpha_{i,j} K(\mathbf{x}_i, \mathbf{x}) \text{ for } j = 1, 2, \ldots, N$$

KPCA algorithm

1. Choose a kernel function – $K(x, x')$
   (ex: gaussian, polynomial, sigmoid, cosine)

2. Compute the kernel (similarity) matrix K for all data points

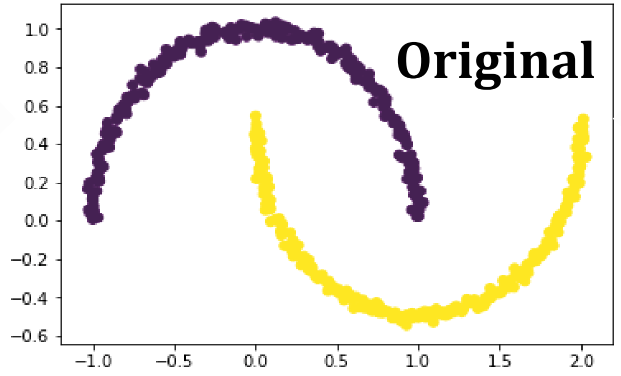
3. Centralize the kernel matrix: $\widetilde{K} = K - \mathbf{1_N}K - K\mathbf{1_N} + \mathbf{1_N}K\mathbf{1_N}$

4. Solve the eigenvalue equation: $\widetilde{K}\alpha_i = \lambda_i \alpha_i$

5. Select the n largest eigenvalues to obtain the PCs in the feature space

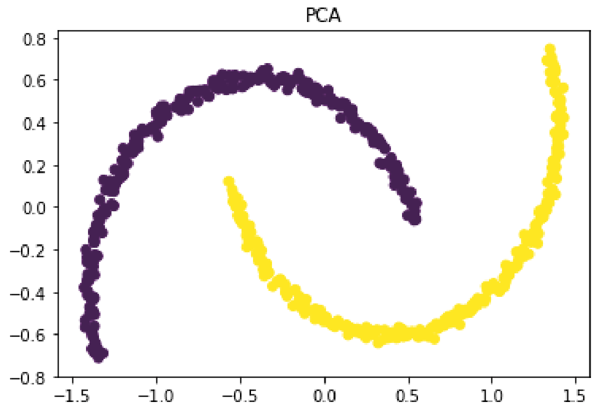
$$y_j = \sum_{i=1}^{d} \alpha_{i,j} K(\mathbf{x}_i, \mathbf{x}) \text{ for } j = 1, 2, \ldots, N$$

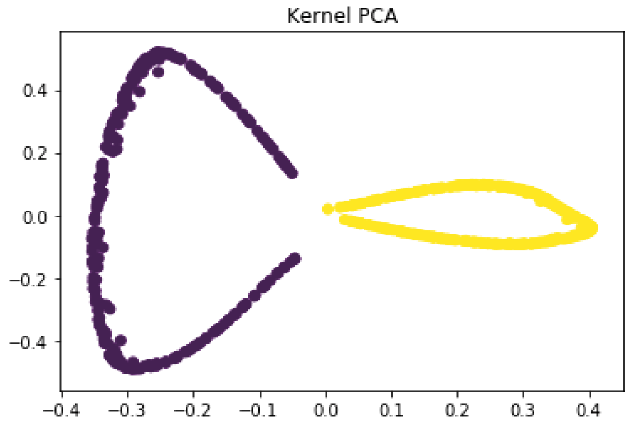
PCA vs KPCA: python example

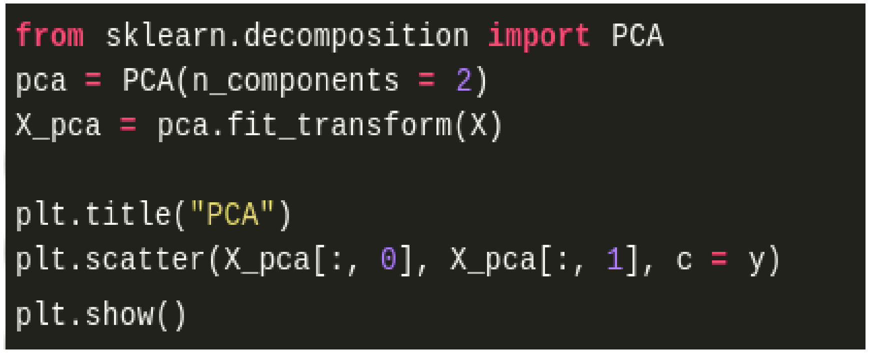


Linear PCA

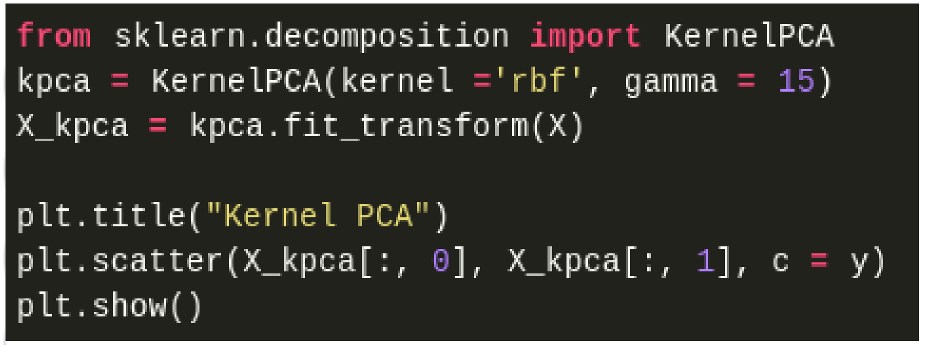
Kernel PCA





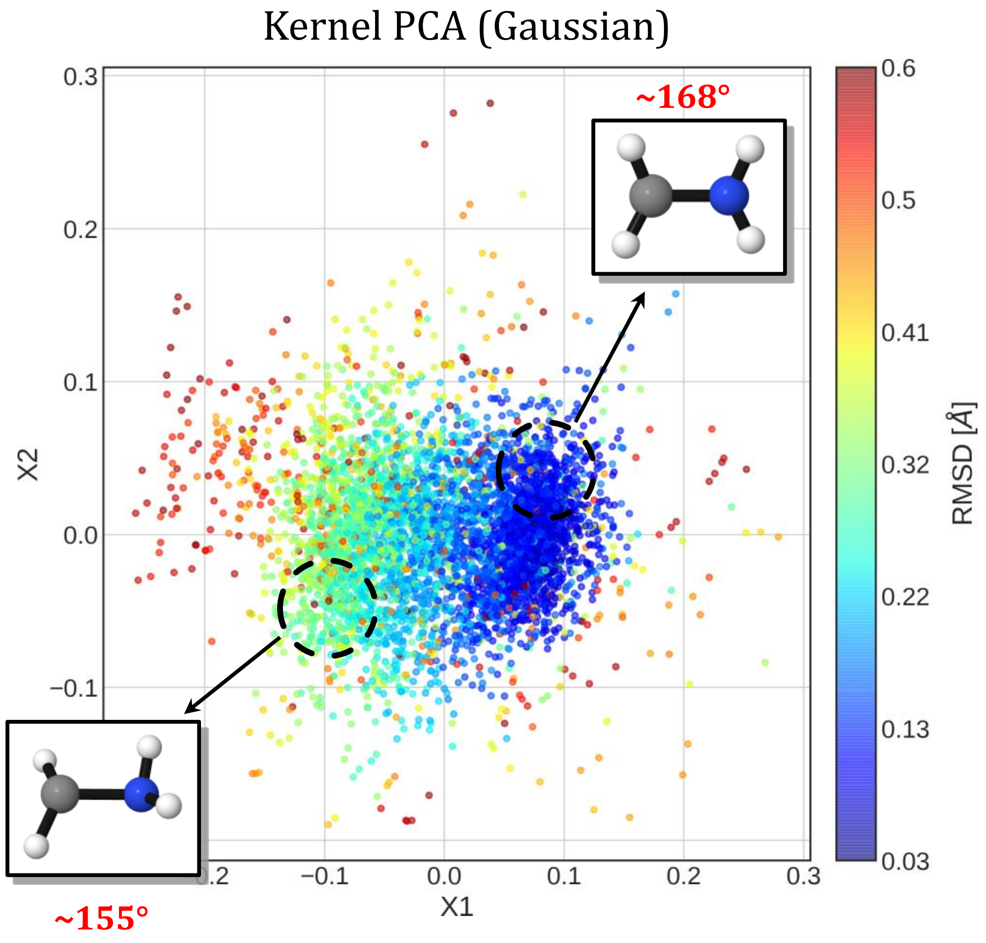


```python
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_pca = pca.fit_transform(X)

plt.title("PCA")
plt.scatter(X_pca[:, 0], X_pca[:, 1], c = y)
plt.show()
```



```python
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(kernel ='rbf', gamma = 15)
X_kpca = kpca.fit_transform(X)

plt.title("Kernel PCA")
plt.scatter(X_kpca[:, 0], X_kpca[:, 1], c = y)
plt.show()
```

KPCA application: molecular dynamics



Most of geometry changes occur along $X1$

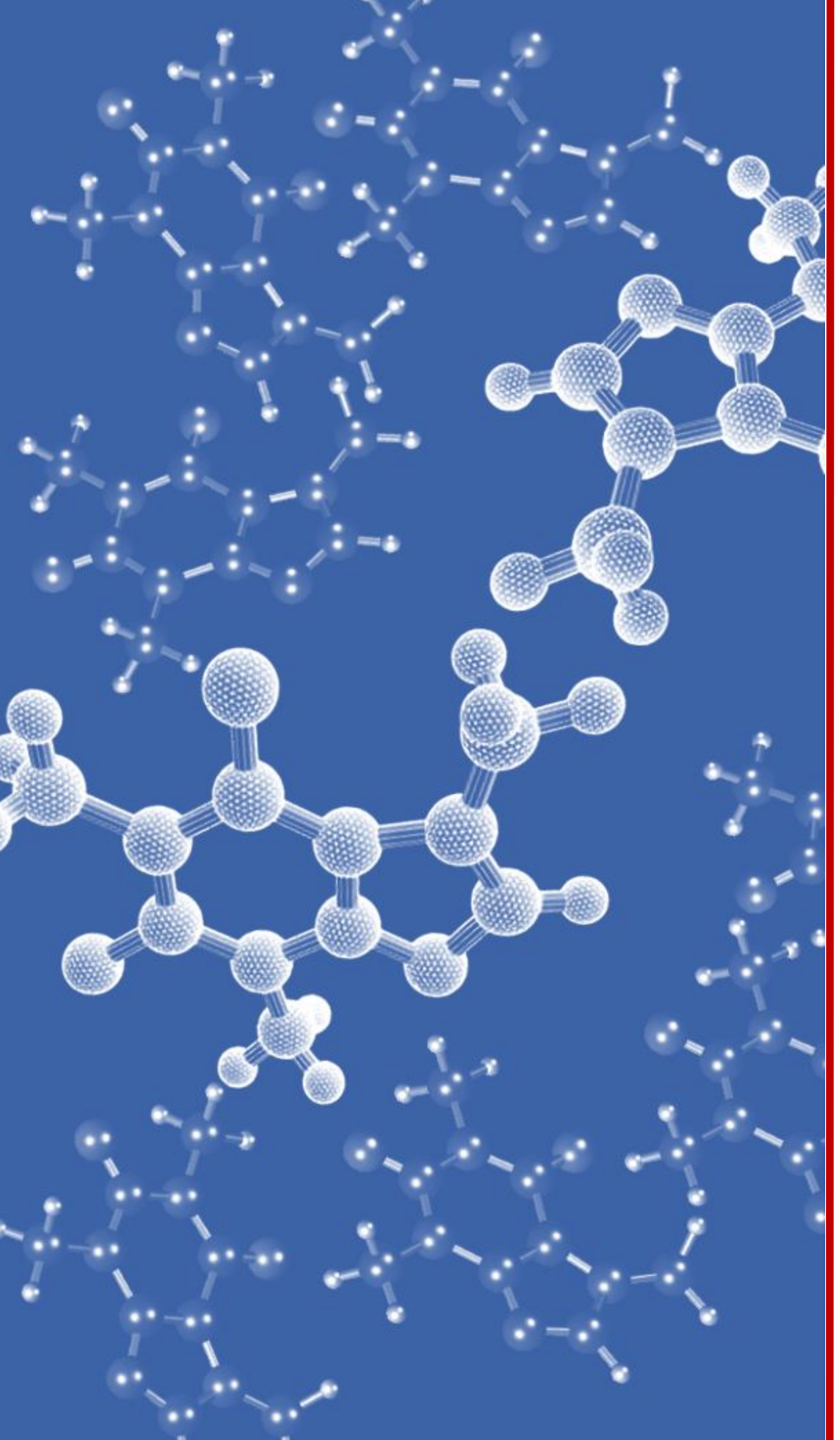
Blue cluster: similar to the S0 equilibrium geometry (small RMSD)

Cyan cluster: rotated NH2 group (near conical intersection?)

Red points: highly distorted or broken geometries

K-PCA:

- Take nonlinearities into account
- Identify reaction paths



Part 2:
Clustering Methods

Clustering methods

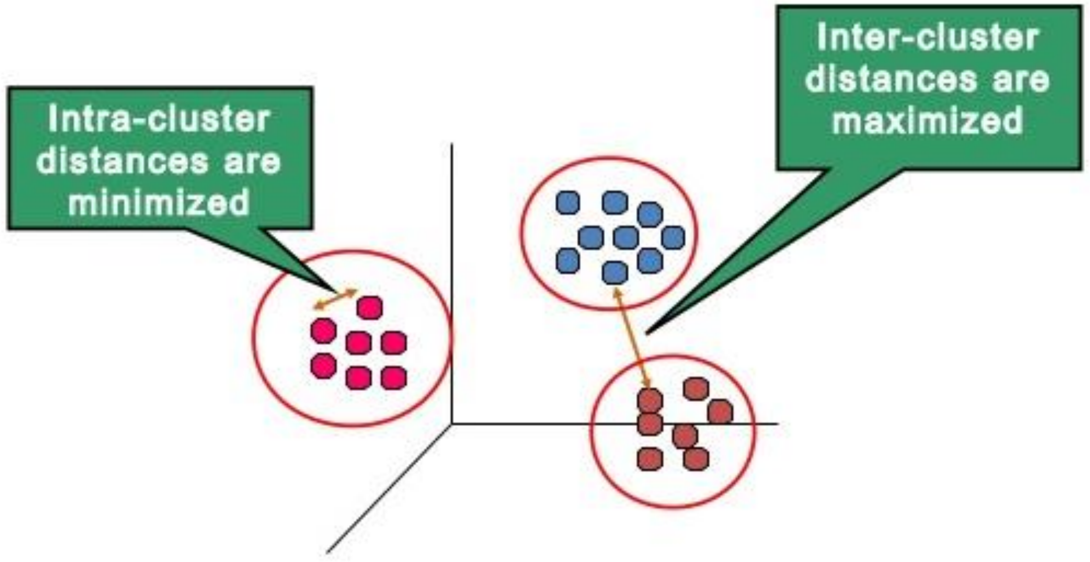
How to **find patterns in data** with unknown labels?

Task: group data by similarity

Clustering methods

“Finding groups of objects such that the objects within a group **be similar (or related)** to one another and **different from (or unrelated to)** the objects in other groups.”

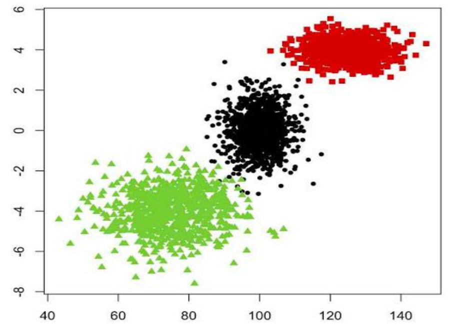
— Page 490, Tan et al, Introduction to Data Mining, 1st edition, 2006.


Intra-cluster distances are minimized
Inter-cluster distances are maximized

Examples of clustering methods

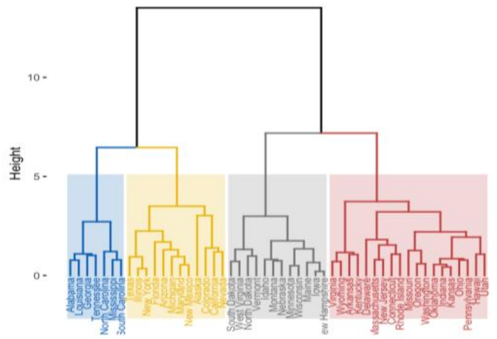
1. Partitional (K-Means, K-Medoids**):**

divides data into **non-overlapping** groups based on distances. Useful for **spherical shape** clusters.



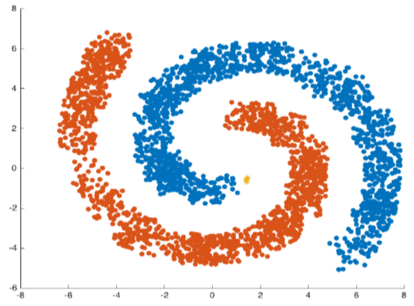
2. Hierarchical (Single-linkage):

determines cluster assignments by building a hierarchy.



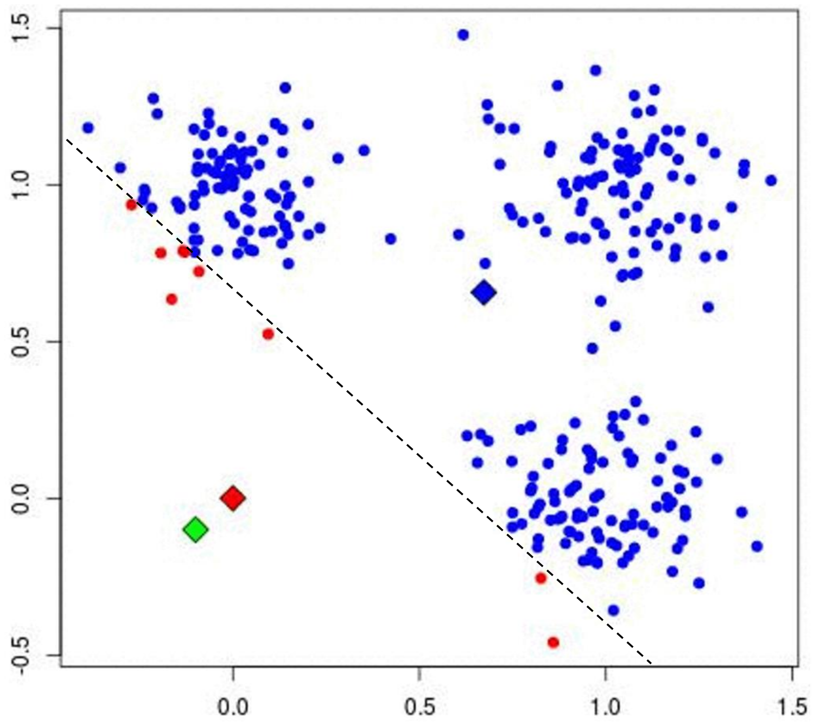
3. Density-based (DBSCAN):

Clusters are assigned to dense regions of data space. Can find arbitrarily shaped clusters. May filter out outliers.



K-Means clustering

Main idea: partition the data into **K clusters** by assigning each observation to the group of points with the **nearest mean** (cluster centroid).



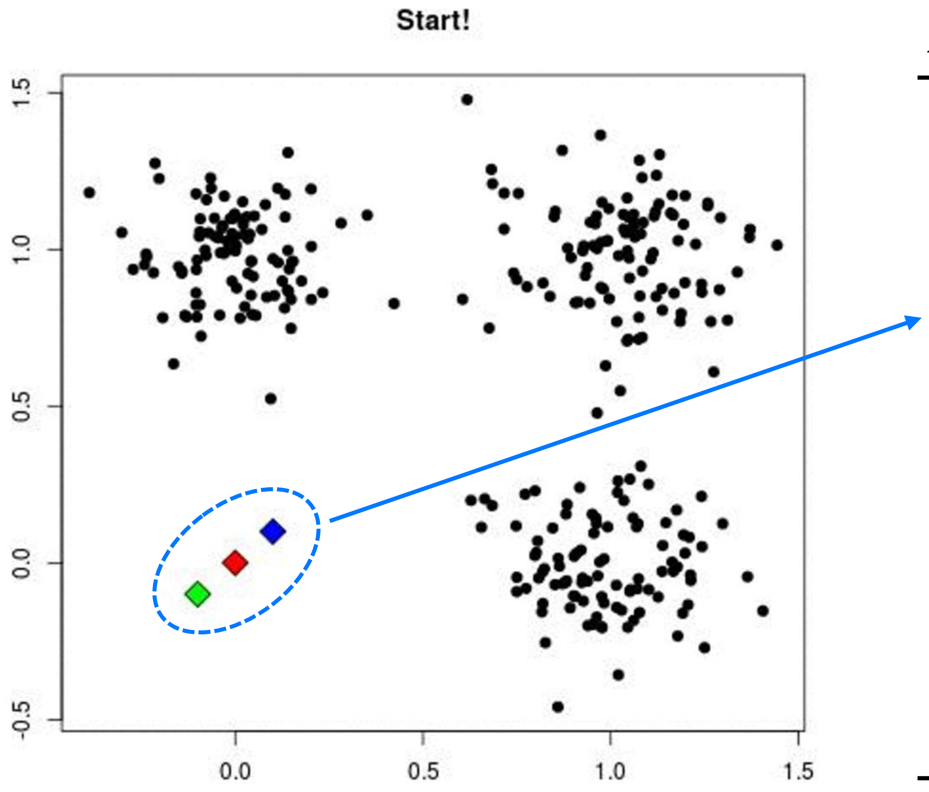
Algorithm

```
Input: data points $\mathbf{X} = [x_1, x_2, ..., x_n], x_i \in \mathbb{R}^n$
Parameter: number of clusters $K$
Output: centroids of the $K$ clusters
1. Initialize the $K$ cluster centers;
while centroids change do
    2. Calculate distances between the centroids
    and all data points;
    3. Assign each data point to its nearest centroid;
    4. Update the centroids using the mean of the
    of the current cluster memberships;
end
```

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-Means clustering

Main idea: partition the data into **K clusters** by assigning each observation to the group of points with the **nearest mean** (cluster centroid).



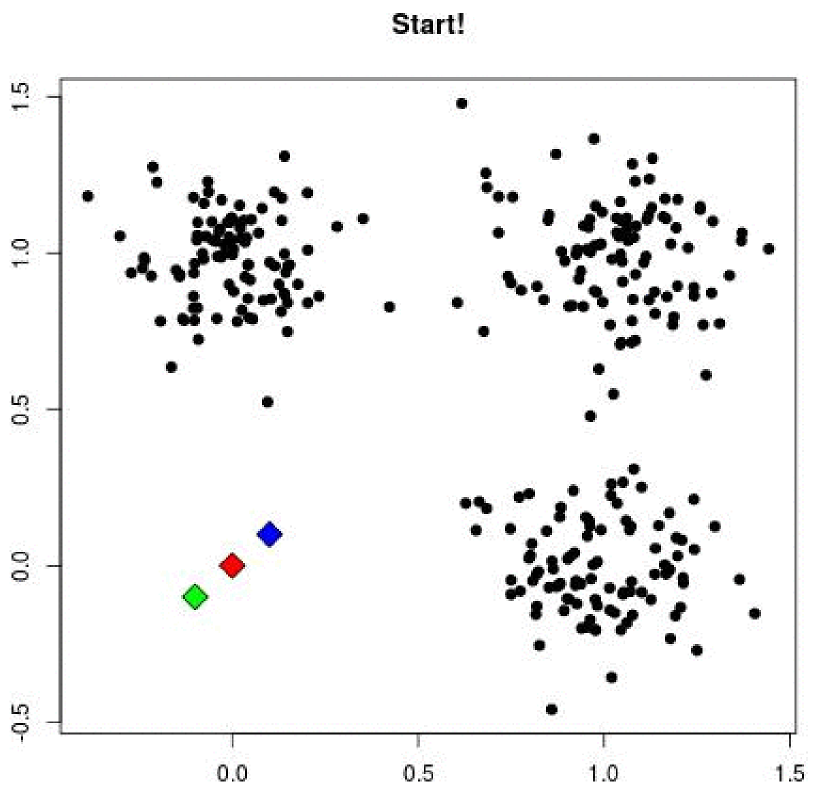
Algorithm

```
Input: data points $\mathbf{X} = [x_1, x_2, ..., x_n], x_i \in \mathbb{R}^n$
Parameter: number of clusters $K$
Output: centroids of the $K$ clusters
1. Initialize the $K$ cluster centers;
while centroids change do
    2. Calculate distances between the centroids
    and all data points;
    3. Assign each data point to its nearest centroid;
    4. Update the centroids using the mean of the
    of the current cluster memberships;
end
```

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

K-Means clustering

Main idea: partition the data into **K clusters** by assigning each observation to the group of points with the **nearest mean** (cluster centroid).



Algorithm

```
Input: data points $\mathbf{X} = [x_1, x_2, ..., x_n], x_i \in \mathbb{R}^n$
Parameter: number of clusters $K$
Output: centroids of the $K$ clusters
1. Initialize the $K$ cluster centers;
while centroids change do
    2. Calculate distances between the centroids
    and all data points;
    3. Assign each data point to its nearest centroid;
    4. Update the centroids using the mean of the
    of the current cluster memberships;
end
```

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

The math behind K-Means

The **objective function** to be minimized is the Sum of Squared Error (SSE), given by:

$$J = \sum_n \sum_k r_{nk} \| x_n - \mu_k \|^2$$

to optimize

$$\text{Where} \begin{cases} x \ \text{ is a sample } n, \ x_n \in \mathbb{R}^D \\ \mu_k \text{ is the cluster center of cluster } k, \mu_k \in \mathbb{R}^D \\ \boldsymbol{r} \ \text{ is the cluster membership, } \mathbf{r} \in \mathbb{R}^{N \times D} \end{cases}$$

The math behind K-Means

$$r_{nk} = \begin{cases} 1, & \text{if } x_n \in \text{cluster } k \\ 0, & otherwise \end{cases} \xrightarrow{\quad} \arg\min_{k'} \left\| x_n - \mu_{k'} \right\|^2 = k$$

$$\mu_k^* = \arg\min_{\mu_k} J \xrightarrow{\quad}$$ **optimal cluster center**

$$\frac{\delta J}{\delta \mu_k} = \frac{\delta[\sum_n \sum_k r_{nk} \| x_n - \mu_k \|^2]}{\delta_{\mu_k}} = 0$$

$$\sum_n r_{nk} \times 2(x_n - \mu_k)(-1) = 0$$

$$\sum_n 2 r_{nk} \mu_k - \sum_n 2 r_{nk} x_n = 0$$

$$\sum_n r_{nk} \mu_k - \sum_n r_{nk} x_n = 0$$

$$\mu_k \sum_n r_{nk} - \sum_n r_{nk} x_n = 0$$

$$\boxed{\mu_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} x_n}$$

Centroid definition

K-Means: *Pros and Cons*

Simple, intuitive and easy to implement

K-means works well when the data form **compact clouds** with globular shapes, and **well separated** from one another

Number of clusters should be provided by the user ($K = ?$)

└→ **Solution:** elbow method

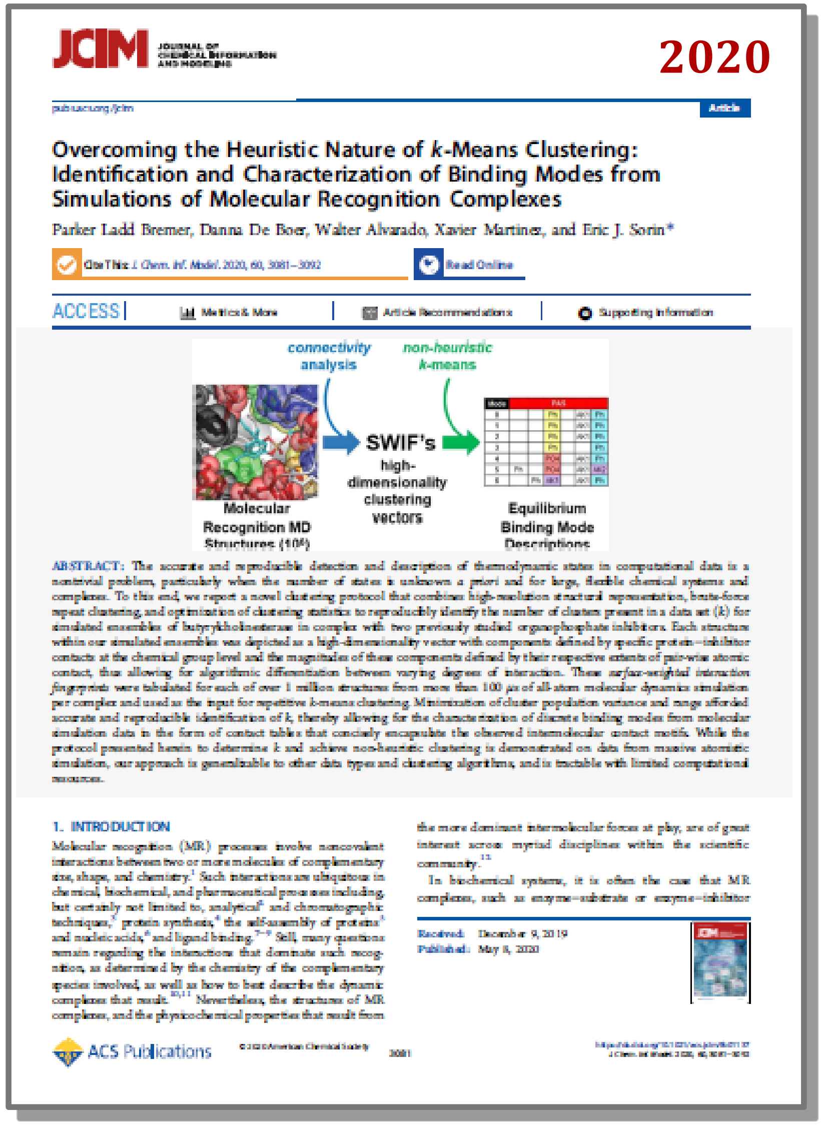
Sensitive to outlier points, which can affect the mean values significantly

└→ **Solution:** data preprocessing!

Sensitive to initialization: clustering results may vary significantly with the initial guess for the cluster centers

└→ **Solution:** multiple executions with random initializations

K-Means in molecular simulations



JOURNAL OF CHEMICAL INFORMATION AND MODELING

2020

pubs.acs.org/jcim

Article

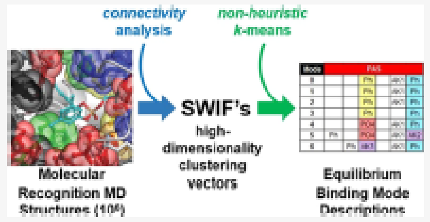
Overcoming the Heuristic Nature of *k*-Means Clustering: Identification and Characterization of Binding Modes from Simulations of Molecular Recognition Complexes

Parker Ladd Bremer, Danna De Boer, Walter Alvarado, Xavier Martinez, and Eric J. Sorin*

Cite This: *J. Chem. Inf. Model.* 2020, 60, 3081−3092

Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information


connectivity analysis
non-heuristic k-means
SWIF's
high-dimensionality clustering vectors
Molecular Recognition MD Structures (10⁶)
Equilibrium Binding Mode Descriptions

ABSTRACT: The accurate and reproducible detection and description of thermodynamic states in computational data is a nontrivial problem, particularly when the number of states is unknown a priori and for large, flexible chemical systems and complexes. To this end, we report a novel clustering protocol that combines high-resolution structural representation, brute-force repeat clustering, and optimization of clustering statistics to reproducibly identify the number of clusters present in a data set (k) for simulated ensembles of butyrylcholinesterase in complex with two previously studied organophosphate inhibitors. Each structure within our simulated ensembles was depicted as a high-dimensionality vector with components defined by specific protein−inhibitor contacts at the chemical group level and the magnitudes of these components defined by their respective extents of pair-wise atomic contact, thus allowing for algorithmic differentiation between varying degrees of interaction. These *surface-weighted interaction fingerprints* were tabulated for each of over 1 million structures from more than 100 μs of all-atom molecular dynamics simulation per complex and used as the input for repetitive *k*-means clustering. Minimization of cluster population variance and range afforded accurate and reproducible identification of k, thereby allowing for the characterization of discrete binding modes from molecular simulation data in the form of contact tables that concisely encapsulate the observed intermolecular contact motifs. While the protocol presented herein to determine k and achieve non-heuristic clustering is demonstrated on data from massive atomistic simulation, our approach is generalizable to other data types and clustering algorithms, and is tractable with limited computational resources.

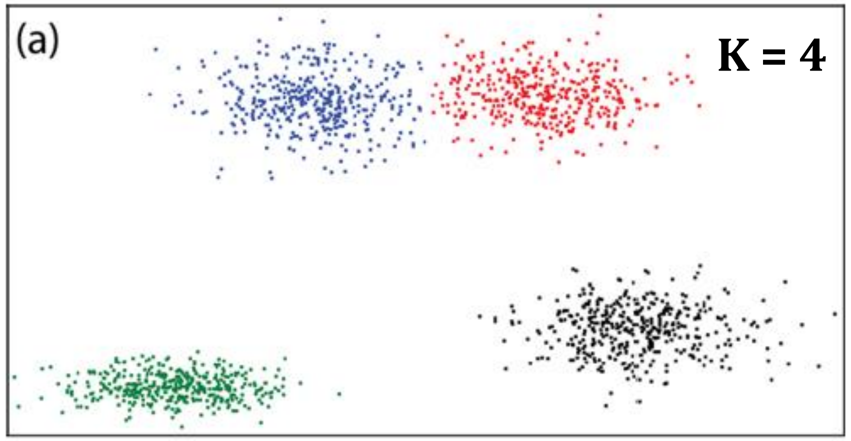
1. INTRODUCTION

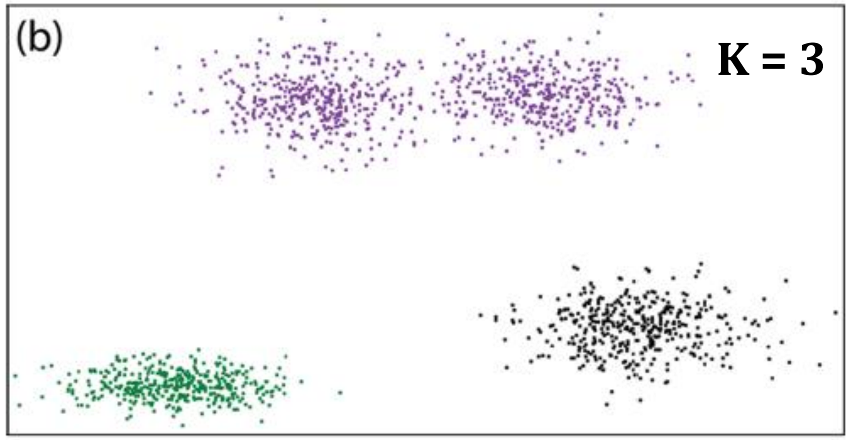
Molecular recognition (MR) processes involve noncovalent interactions between two or more molecules of complementary size, shape, and chemistry.[1] Such interactions are ubiquitous in chemical, biochemical, and pharmaceutical processes including, but certainly not limited to, analytical[2] and chromatographic techniques,[3] protein synthesis,[4] the self-assembly of proteins[5] and nucleic acids,[6] and ligand binding.[7−9] Still, many questions remain regarding the interactions that dominate such recognition, as determined by the chemistry of the complementary species involved, as well as how to best describe the dynamic complexes that result.[10,11] Nevertheless, the structures of MR complexes, and the physicochemical properties that result from the more dominant intermolecular forces at play, are of great interest across myriad disciplines within the scientific community.[12]

In biochemical systems, it is often the case that MR complexes, such as enzyme−substrate or enzyme−inhibitor

Received: December 9, 2019
Published: May 8, 2020

How to **automatically identify** different types of **protein-inhibitor complexes** from **MD data**? Ans: **K-Means!**





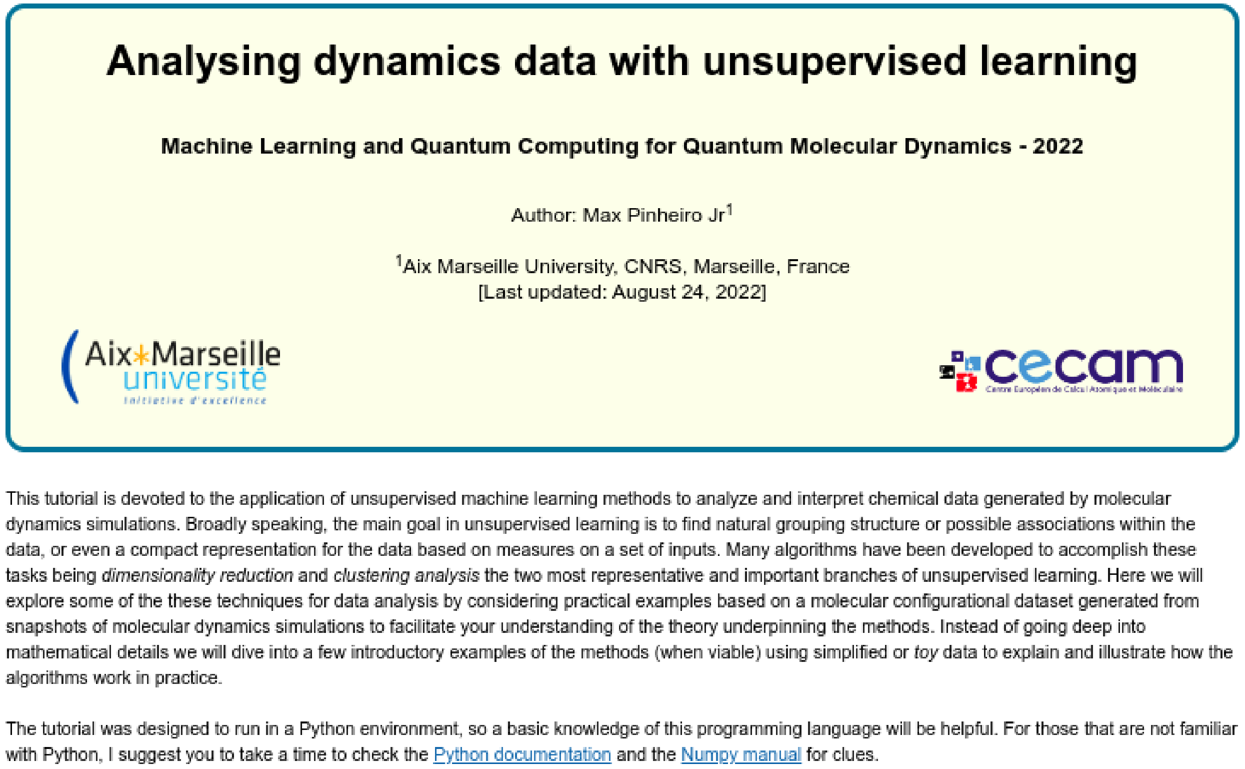
Take-home messages

- ✓ Unsupervised learning is quite challenging
  - input = data + some prior knowledge + intuition.
- ✓ Unsupervised ≈ exploratory data analysis → **try to feel the different “flavors” of your data!**
- ✓ Unsupervised can be useful for labeled data. Validate your analysis!
- ✓ Dimensionality reduction: projected data does not always reflect the relationships/distance of the original data space.
- ✓ Data quality for clustering:
  - choose an adequate representation for the chemical system
  - invest time in preprocessing
  - which distance metric should I use?

KEEP CALM
WE ARE ALMOST THERE
imgflip.com

Hands-on: now it is time to practice

Tutorial in Python: github.com/maxjr82/CECAM-MLQCDyn



Analysing dynamics data with unsupervised learning

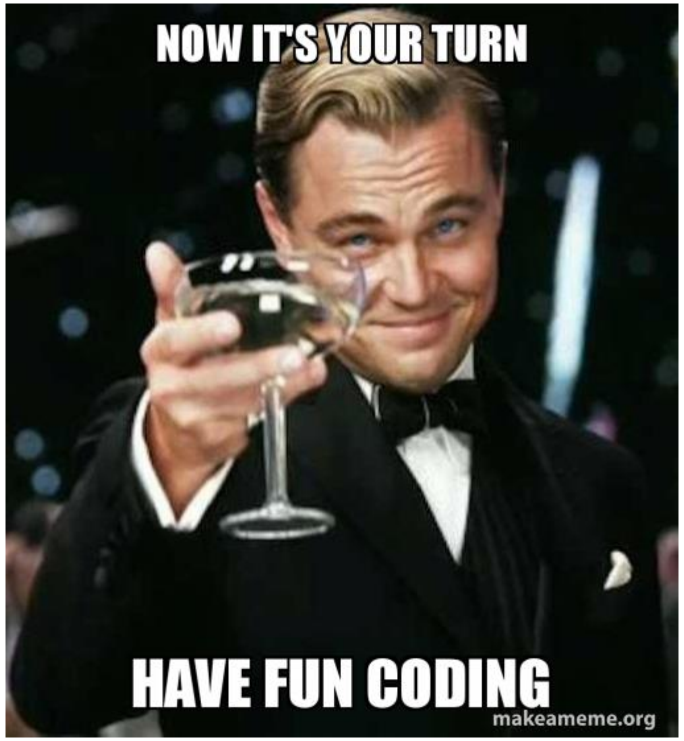
Machine Learning and Quantum Computing for Quantum Molecular Dynamics - 2022

Author: Max Pinheiro Jr[1]

[1]Aix Marseille University, CNRS, Marseille, France
[Last updated: August 24, 2022]

This tutorial is devoted to the application of unsupervised machine learning methods to analyze and interpret chemical data generated by molecular dynamics simulations. Broadly speaking, the main goal in unsupervised learning is to find natural grouping structure or possible associations within the data, or even a compact representation for the data based on measures on a set of inputs. Many algorithms have been developed to accomplish these tasks being *dimensionality reduction* and *clustering analysis* the two most representative and important branches of unsupervised learning. Here we will explore some of the these techniques for data analysis by considering practical examples based on a molecular configurational dataset generated from snapshots of molecular dynamics simulations to facilitate your understanding of the theory underpinning the methods. Instead of going deep into mathematical details we will dive into a few introductory examples of the methods (when viable) using simplified or *toy* data to explain and illustrate how the algorithms work in practice.

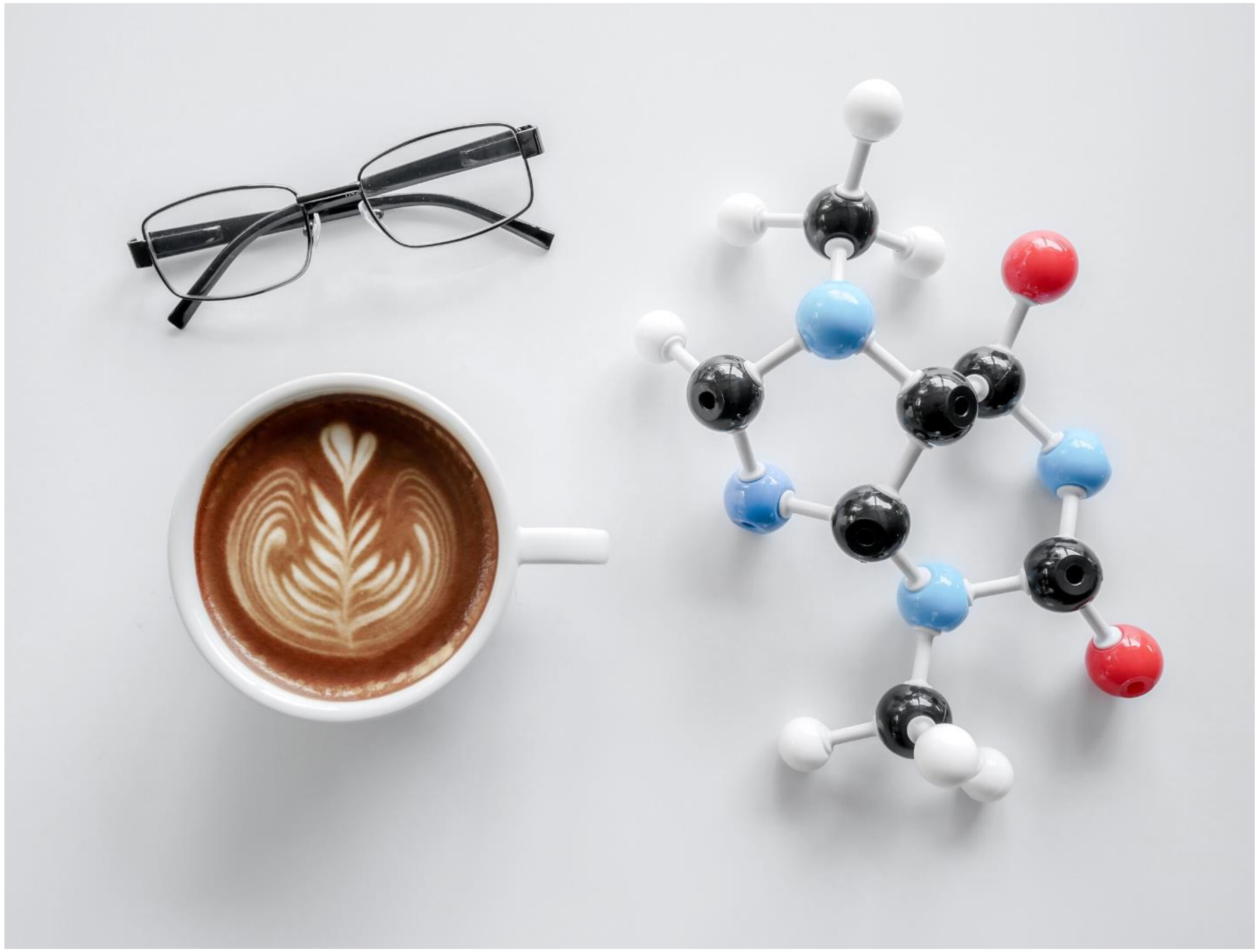
The tutorial was designed to run in a Python environment, so a basic knowledge of this programming language will be helpful. For those that are not familiar with Python, I suggest you to take a time to check the Python documentation and the Numpy manual for clues.



GitHub

Google Colab

Send questions to: maxjr82@gmail.com



Thank you!