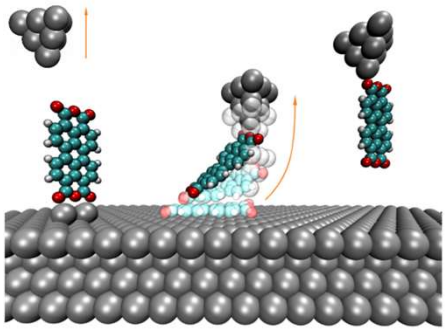
Introduction to Machine Learning

Reinhard J. Maurer
Department of Chemistry & Department of Physics,
University of Warwick

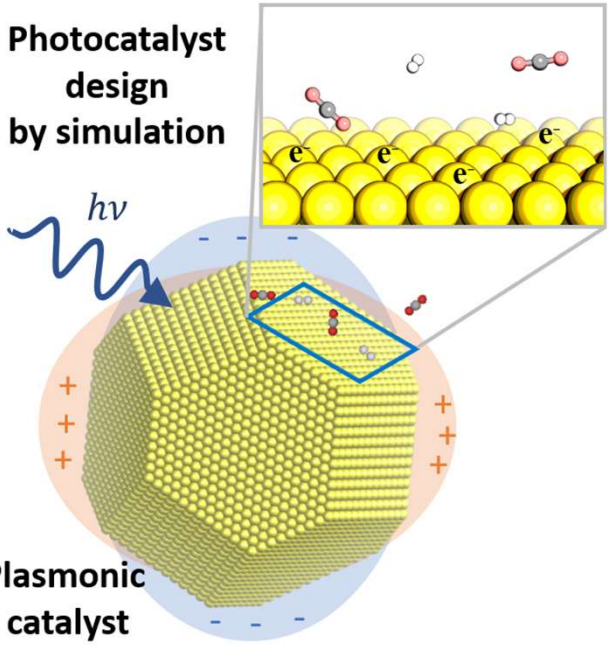
www.warwick.ac.uk/maurergroup

Computational Surface Science

Metastable nanostructures

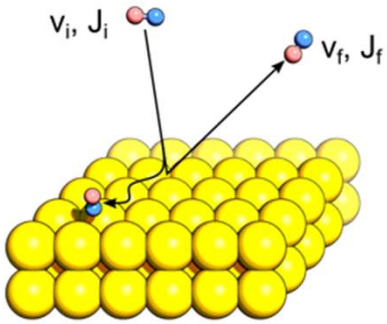


Photocatalyst design by simulation



Plasmonic catalyst

Surface Chemistry



UK Research and Innovation

Engineering and Physical Sciences Research Council

LEVERHULME TRUST

diamond

Our approach

FHI-aims
The ab initio materials simulation package

Electronic Structure Theory

Machine Learning of Electronic Structure

Coupling of Light and Electrons

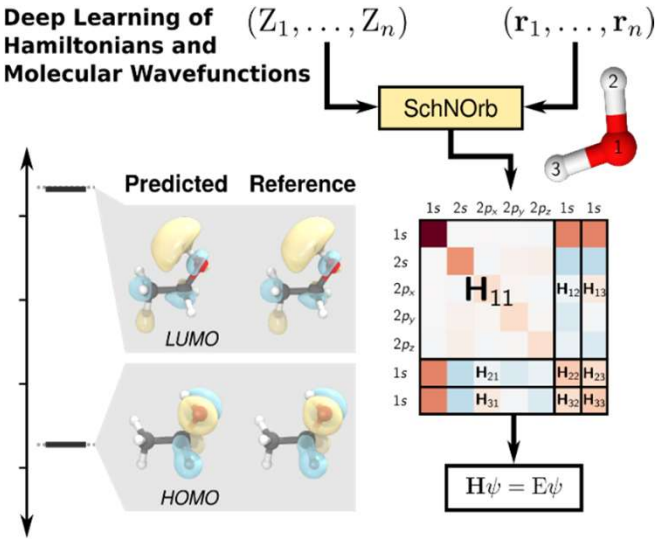
Nonadiabatic Molecular Dynamics

NQCDynamics.jl

Develop machine learning surrogate models of electronic structure

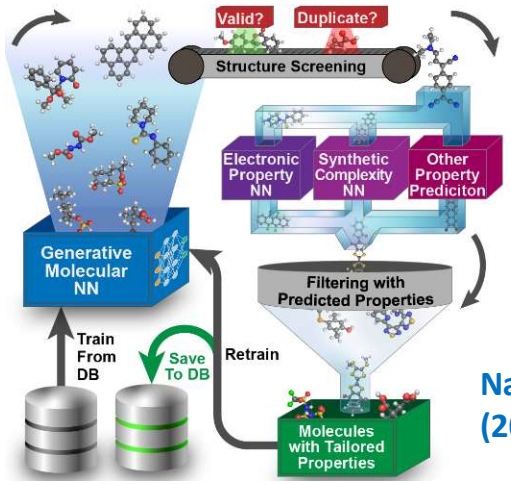
Machine Learning of Electronic Structure

Deep Learning of Hamiltonians and Molecular Wavefunctions



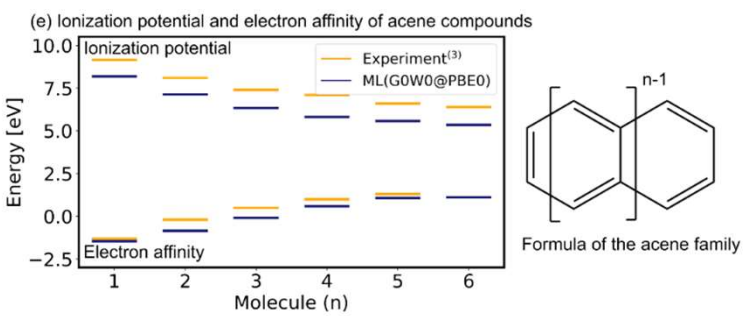
Nature Commun 10, 5024 (2019)

Inverse property-driven design of electronic properties



Nature Comp. Sci. (2023)

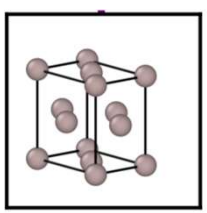
Deep learning of molecular spectroscopy

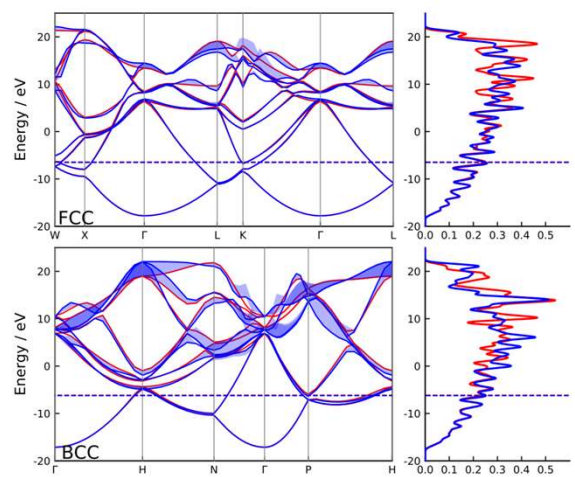


Chem. Sci. 12, 10755-10764 (2021)

Machine learning of electronic Hamiltonians

bulk Al





DFT
ML

npj Computational Materials (2022)

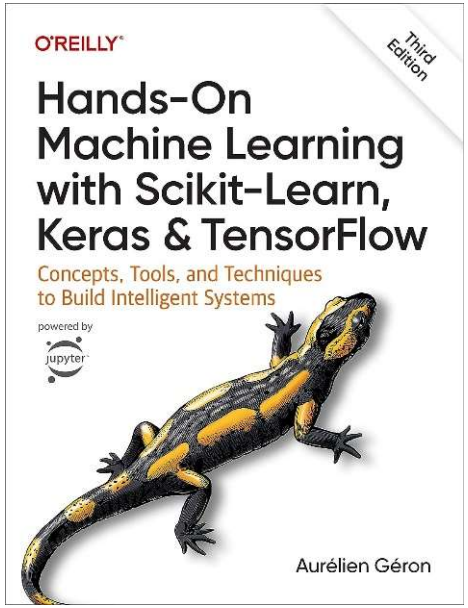
Goal of this lecture

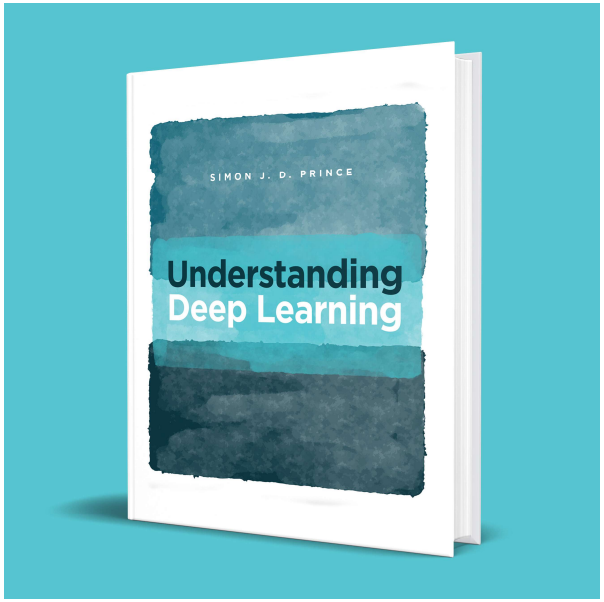
After this lecture (and accompanying workshop), you should ideally

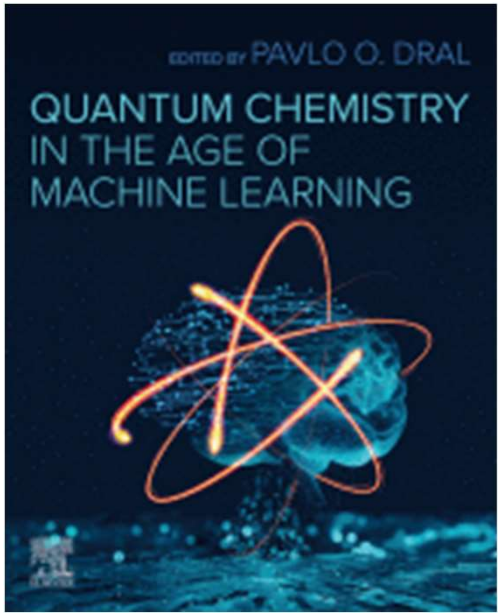
1. Know the **role of machine learning** in the computational physical sciences
2. Understand the **basic terminology** of machine learning ("Slang busting")
3. Have an **overview of methodologies** and how they connect
4. Understand how to approach **a typical machine learning workflow**
5. Know how to **prepare and analyse datasets**
6. Be able to validate and optimise models with **cross-validation**
7. Know basic approaches to **featurisation and representation** in chemistry
8. Know how to evaluate and assess prediction errors and uncertainties

Resources

The Internet







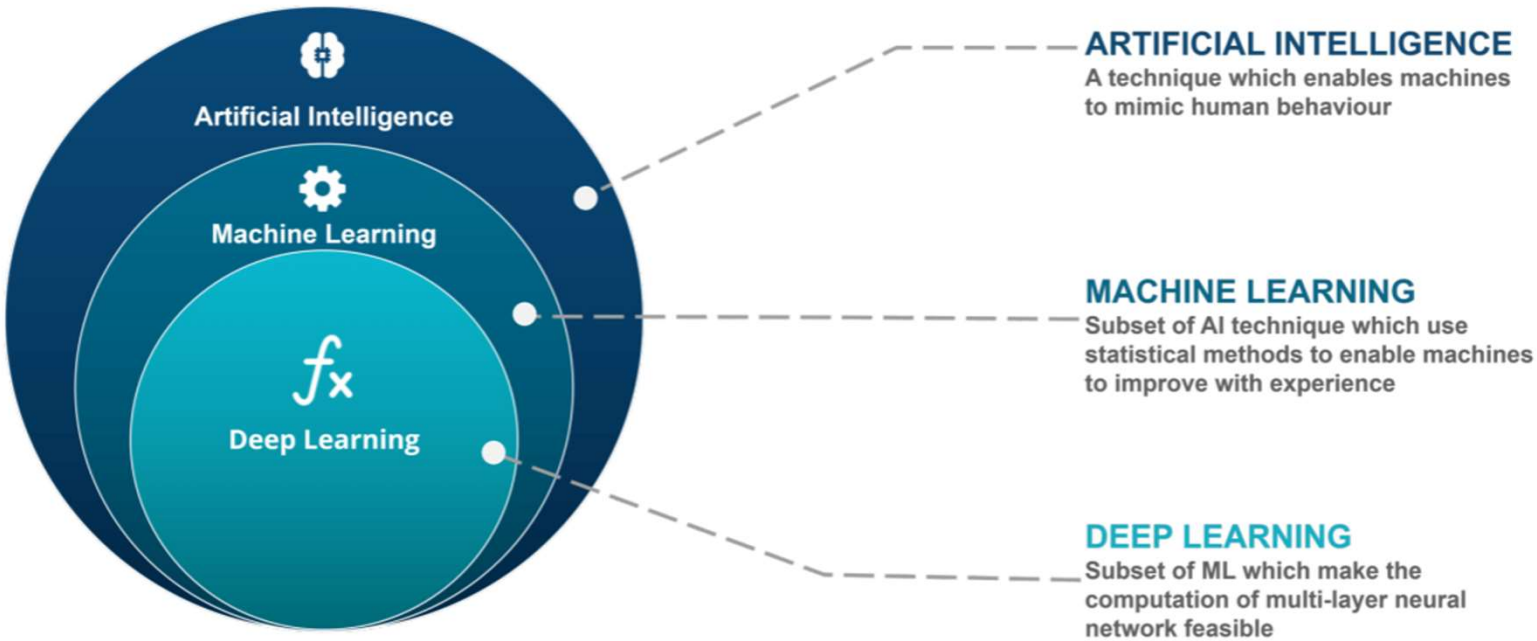
Perspective on integrating machine learning into computational chemistry and materials science

Cite as: J. Chem. Phys. 154, 230903 (2021); doi: 10.1063/5.0047760

Agenda

1. What is ML?
2. Basic Definitions
3. Data Representations and Features / Descriptors
4. Types of ML methods
5. Putting it all together
6. A research example

What is ML and how can it be useful?



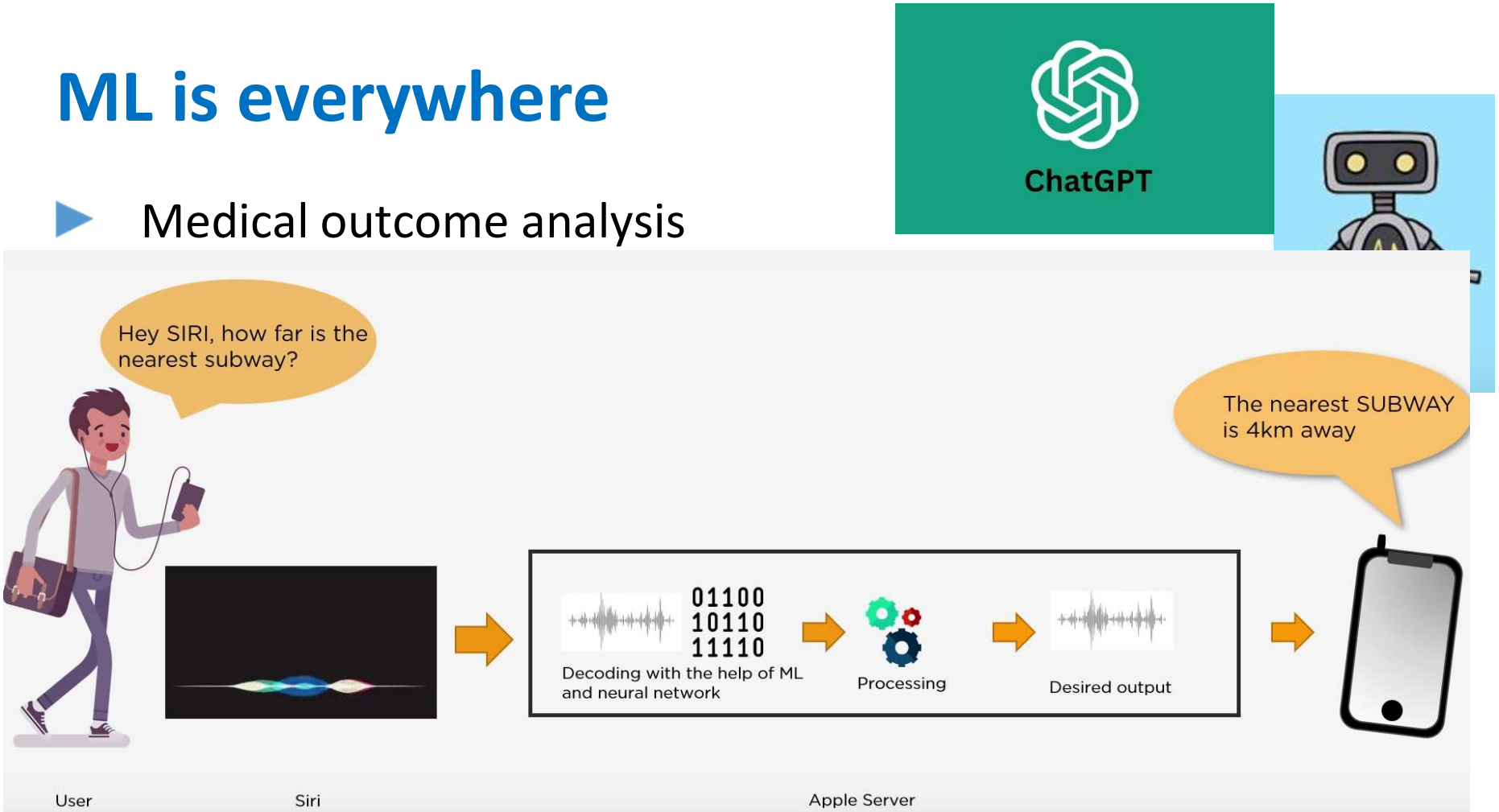
Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.

Arthur Samuel

Learning from Data

ML is everywhere

- Medical outcome analysis



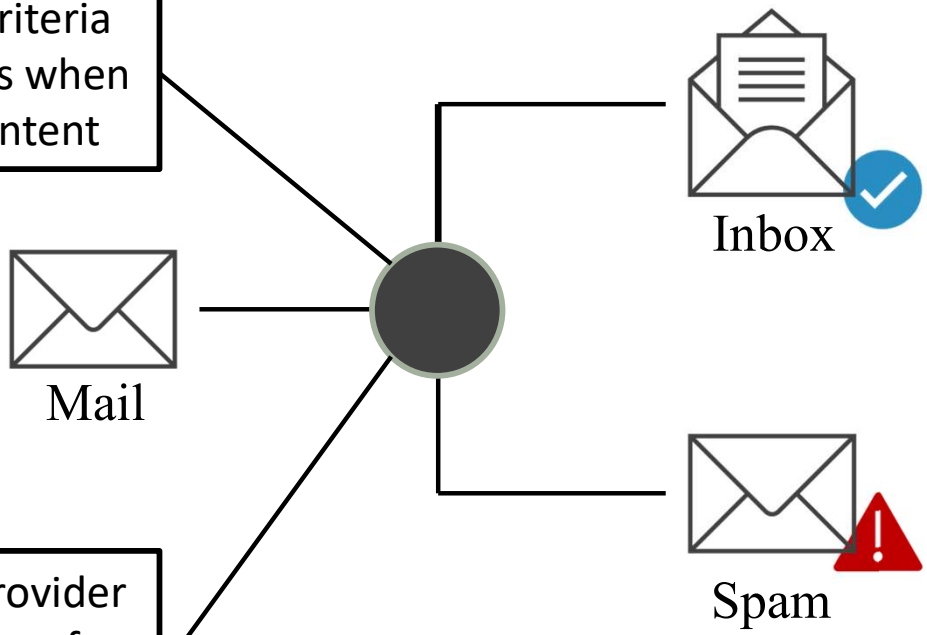
Commonly used in Chemistry and Materials Science research

The ML paradigm and your email spam filter

Classical Computing Paradigm

Determines category

Predefined set of criteria that the code checks when analyzing email content



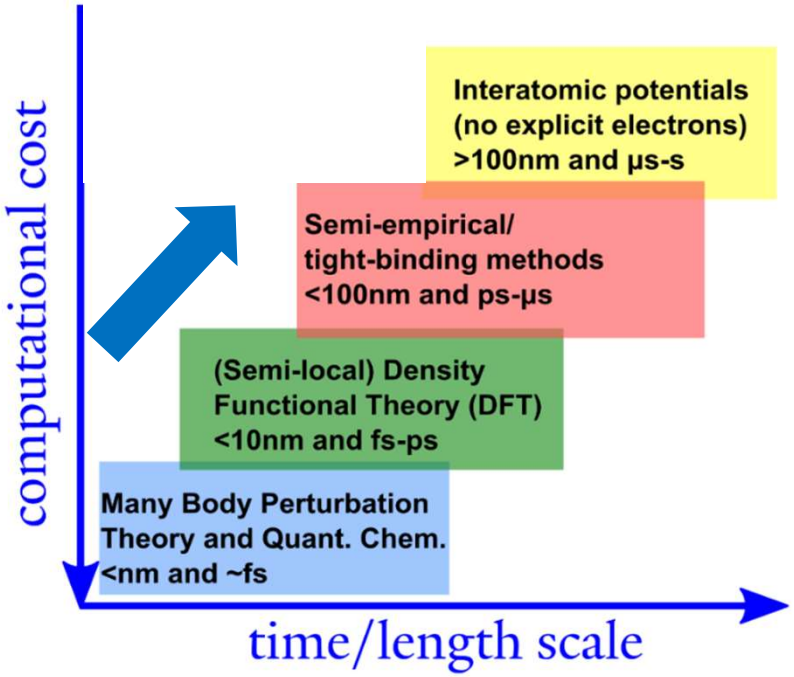
Machine Learning Paradigm

Predicts category

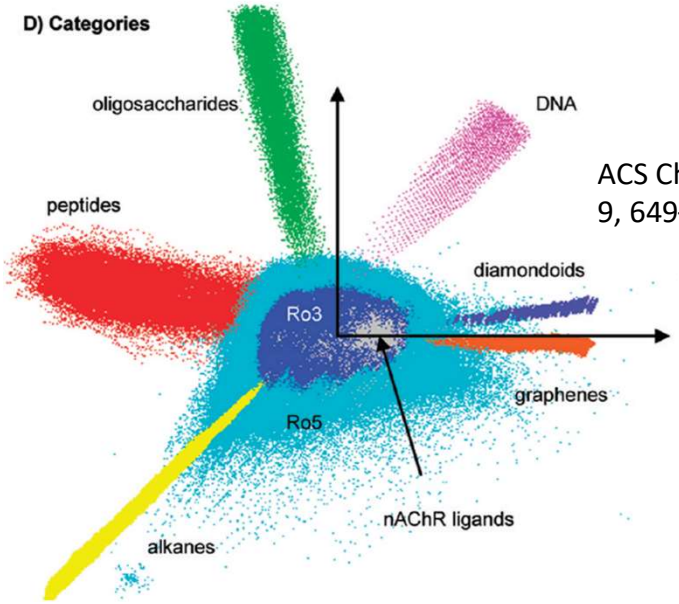
Learns from user/provider data what features of emails are most likely associated with spam

Challenges in Chemistry & Materials Science

Accurate Quantum Chemistry is slow
Fast Quantum Chemistry is inaccurate



Chemical compound space is bigger than the known universe



ACS Chem. Neurosci. 9, 649–657 (2012)

10^{180} possible molecules 10^{80} atoms in universe

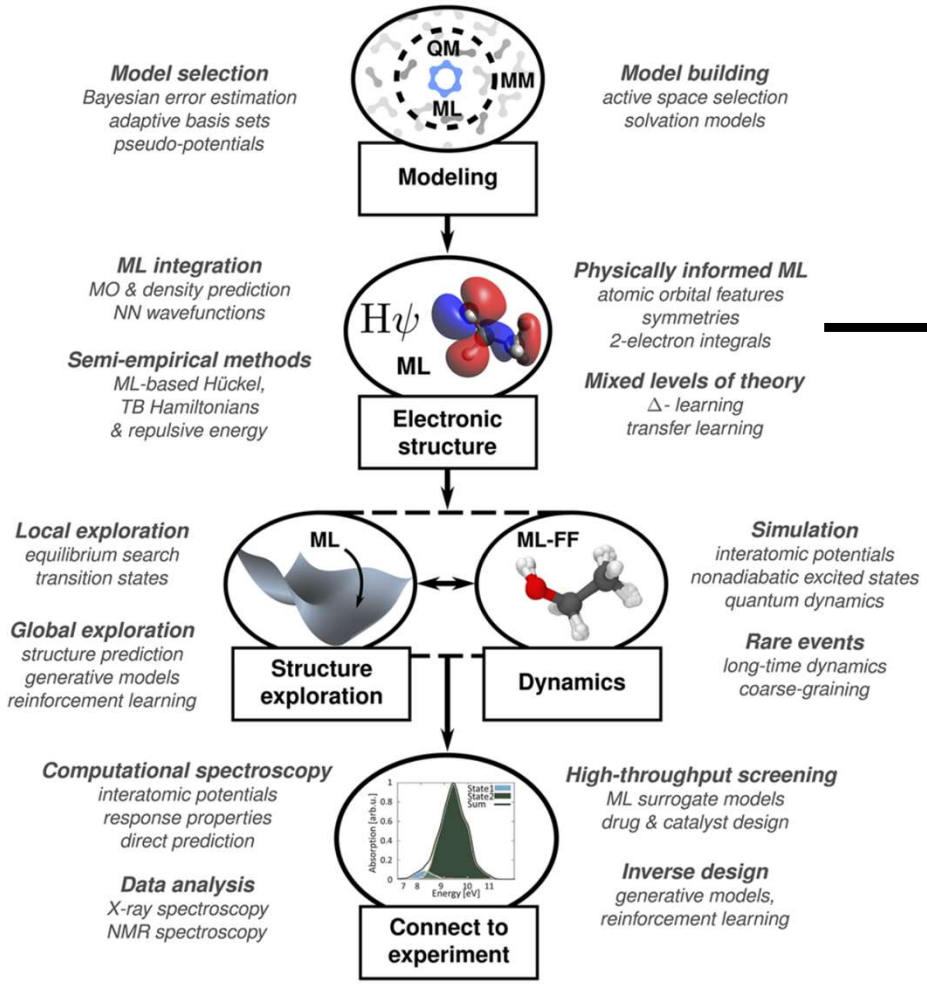
10^{8} known molecules

Machine Learning can help!

Accelerate property prediction

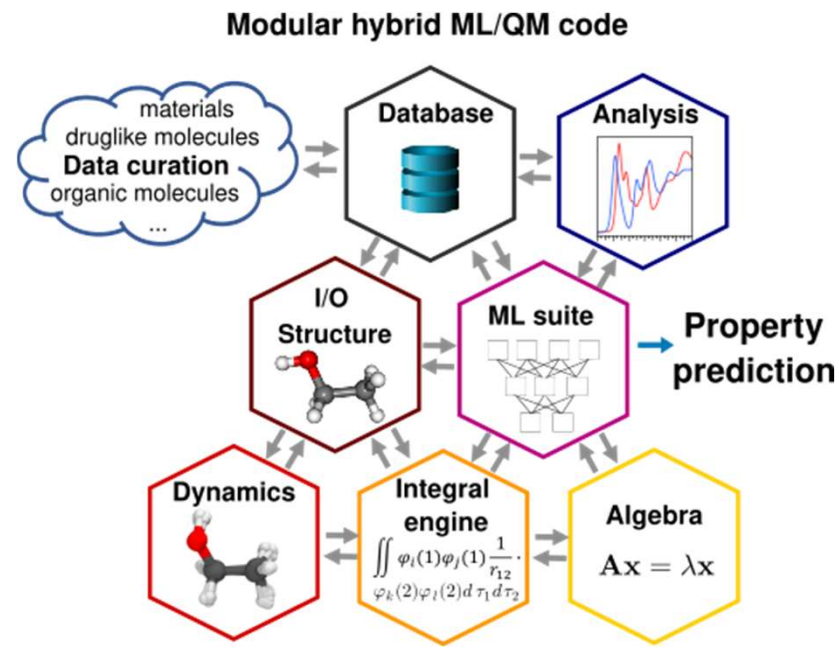
Accelerate materials design

Why should you use ML in your research?



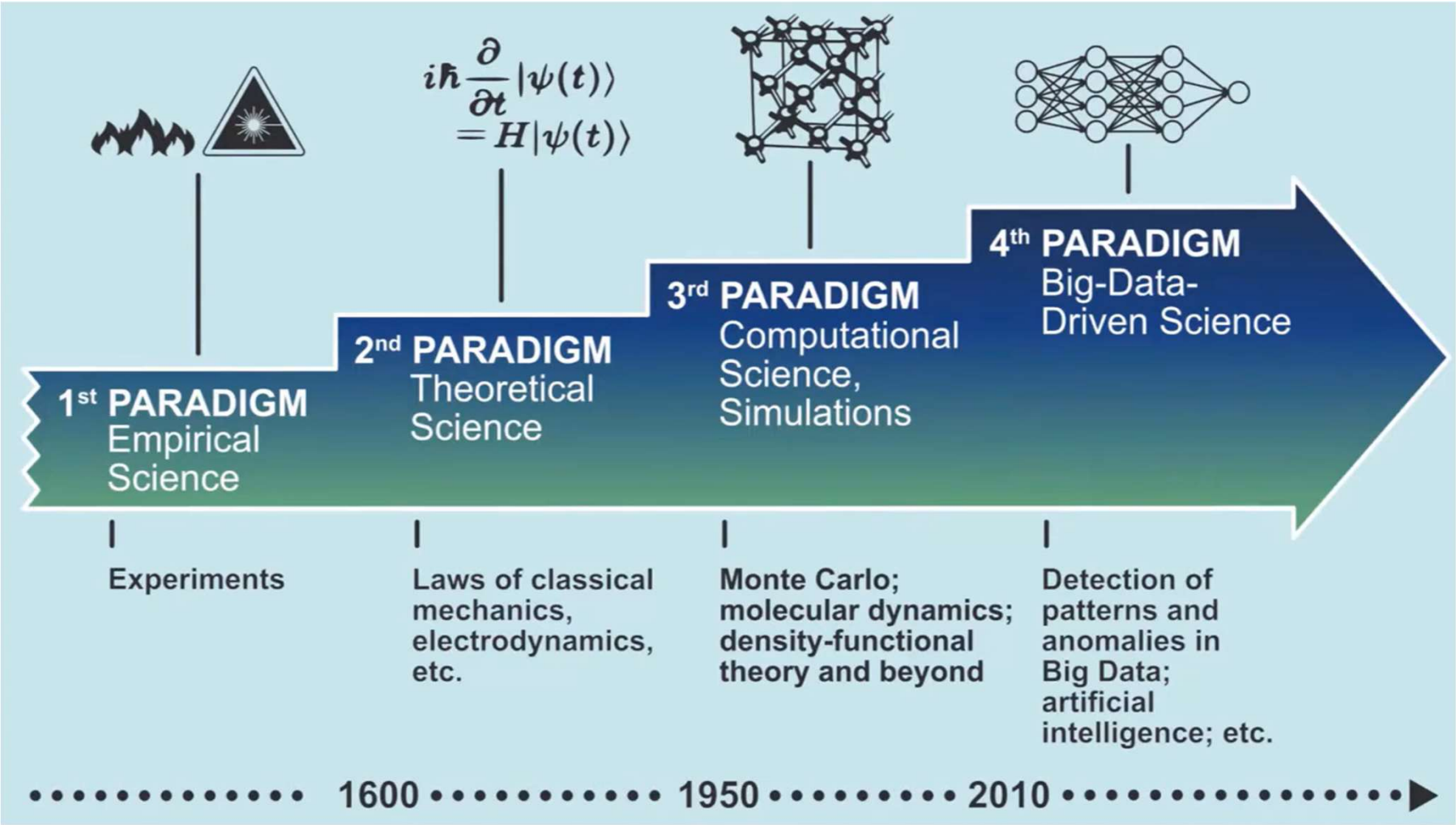
Perspective: J. Chem. Phys. 154, 230903 (2021)

$$\hat{H}\Psi = E\Psi$$



Roadmap on ML in Electronic Structure
IOP Electronic Structure (2022)

The 4th paradigm of science



“Scientific” Machine Learning (SciML)

1. Machine Learning (ML) is an area of Artificial Intelligence (AI) that fits mathematical models to observed data. ML is often applied as a black box, which can limit model transferability.

2. Scientific Computing uses large scale modelling of bottom-up 'real' physics/chemistry, typically through numerical solution of differential equations.
This is accurate and predictive, but often very computationally expensive.

Scientific Machine Learning (SciML) combines the strengths of the two approaches.

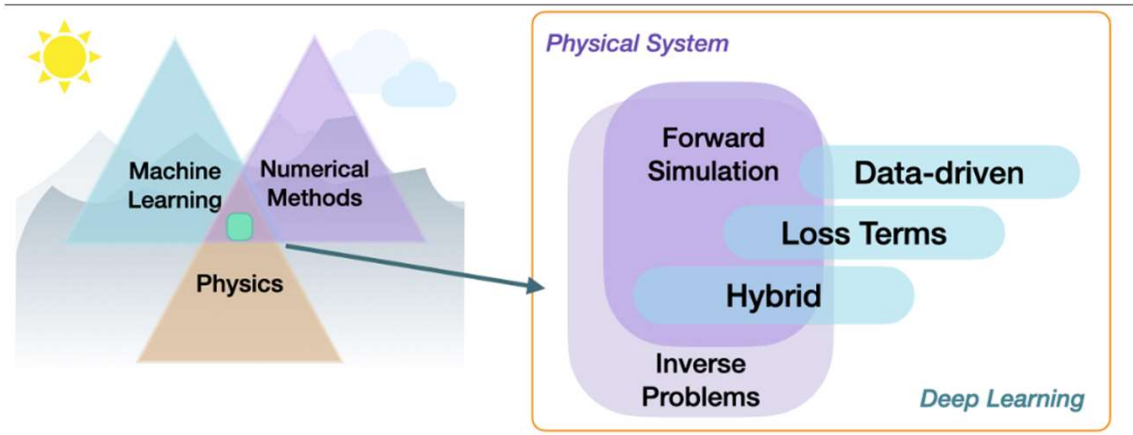
physical laws + data-driven machine learning methods

domain expertise human-interpretable

physics-informed neural networks

limited quality data labels

symmetries and conservation laws

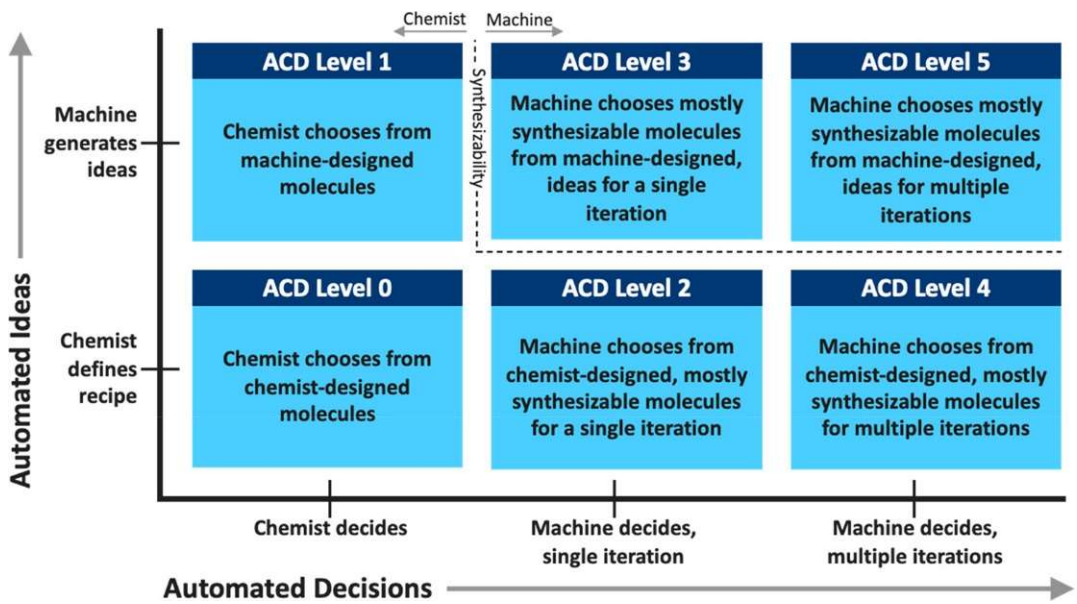


Automated Chemistry by Design (ACD) in Molecular Discovery

Chemistry by design means designing the composition and structure of a molecule or material to achieve specific properties and reactivity

Chemical compound space is bigger than the known universe

Various levels of automation can be imagined

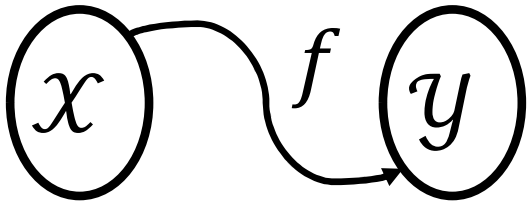


How are new ideas generated?
How are decisions made?

Goldman et al., J. Med. Chem. 65, 10, 7073–7087 (2022)

Basic Definitions

ML Definitions



ML is concerned with algorithms that improve with increasing amount of available data under some performance measure.

Find a predictive function that connects input space $\mathcal{X}$ to a target space $\mathcal{Y}$

$$f\colon \mathcal{X} \to \mathcal{Y}$$

ML focuses on **universal approximators**, able to represent any function with arbitrary accuracy, when given enough training data and parameters.

The functional relationship to be found is specified by choosing a suitable **loss function** $\ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y})$.

If the loss function requires knowledge of targets (labels) $y \in \mathcal{Y}$, we speak of **supervised learning.**

How it works

Training Dataset
data that model sees to learn

1. Provide a training dataset, T
2. The machine is **trained** to capture statistical trends in T and suggests a function f
3. Ideally, $f(x_i) = y_i$ for all values
4. Then, given new (unseen) x' values, the model should **predict** $f(x') = y'$
5. We can validate how much the model has learned with a **test set** of data for which the ground truth is known

Test/Validation Dataset
Data that is unseen to assess learning. Generate by splitting off some data (~20%)

We train a model by minimising risk

The optimal model minimizes the expected risk, $R(f)$, defined as the **expectation value** of $\ell(f(\mathcal{X}), \mathcal{Y})$:

$$R(f) = \langle \ell(f(\mathcal{X}), \mathcal{Y}) \rangle = \int \ell(f(\mathcal{X}), \mathcal{Y}) \ \mathrm{d}\mathcal{P}(\mathcal{X}, \mathcal{Y})$$

We typically don't know the underlying data distribution $\mathcal{P}(\mathcal{X}, \mathcal{Y})$ so we use the **empirical risk** instead, which is the expectation over a finite data set T:

$$R_{\mathrm{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n \in T} \ell(f(x_i), y_i)$$

The ML training process is the process of minimizing R_{emp} or the **loss**.

Different types of loss functions

Different ways to measure distances between the ground truth and the model prediction f by a single number.

L1-norm

$$J(f) = \sum_{i}^{n} |f_i(x_i) - y_i|$$

Least absolute deviations

L2-norm

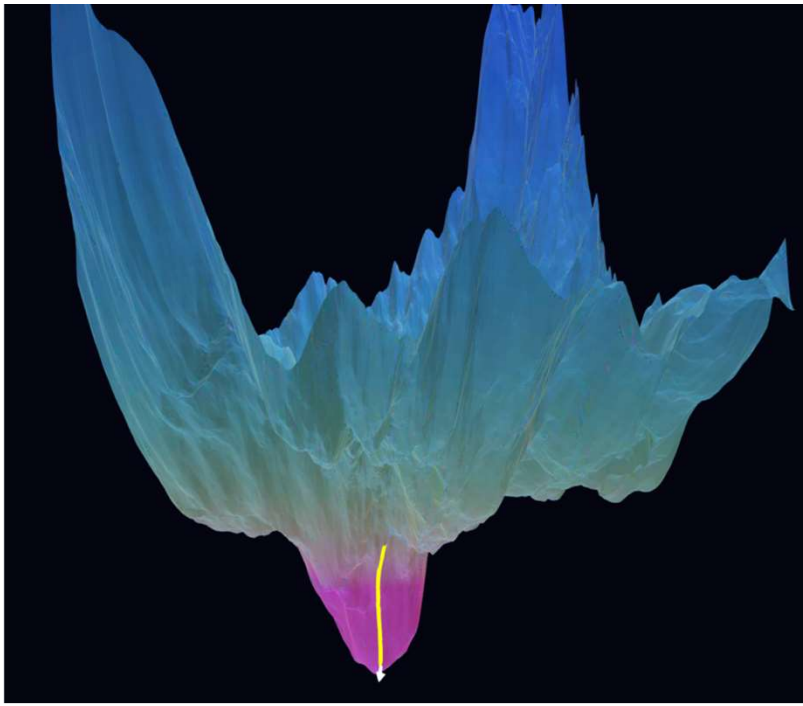
$$J(f) = \sum_{i}^{n} (f_i(x_i) - y_i)^2$$

Least square regression

L1 norm more robust against outliers, but more difficult to optimize

Training and Overfitting

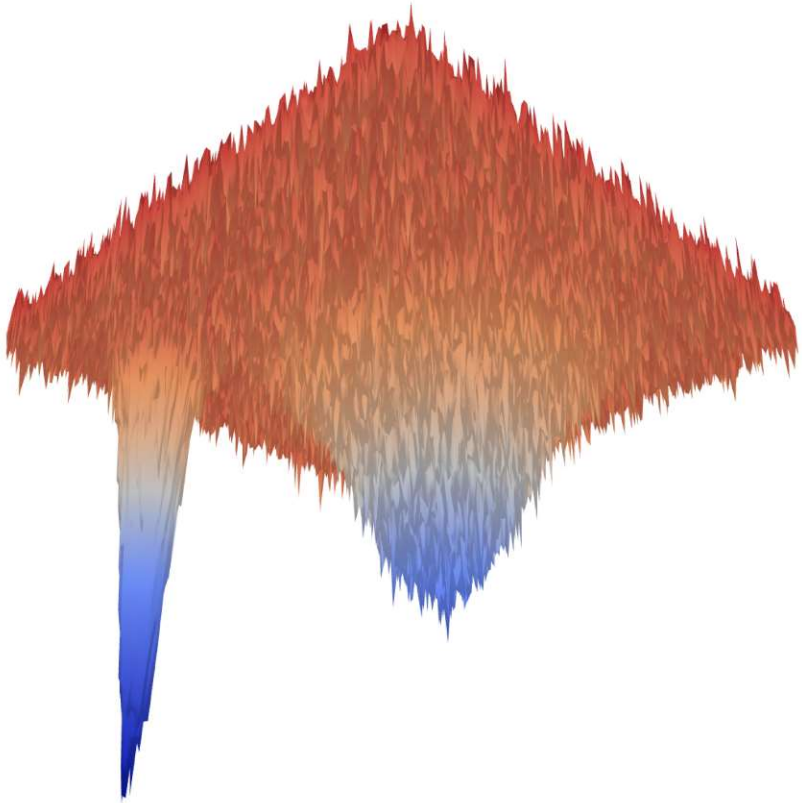
- Unfortunately, $f(x_i) = y_i$ does not hold and there is an error
- Many functions can map $\mathcal{X} \rightarrow \mathcal{Y}$
- Let F be the set of functions that map $\mathcal{X} \rightarrow \mathcal{Y}$
- The accuracy of any $f \in F$ is determined by the loss function $\ell(f(\mathcal{X}), \mathcal{Y})$
- **Training is the process of minimizing the loss e.g. through (stochastic) gradient descent**
- Since the set F is large, we introduce **regularizer** terms to the optimization problem, which punish complex solutions that lead to overfitting



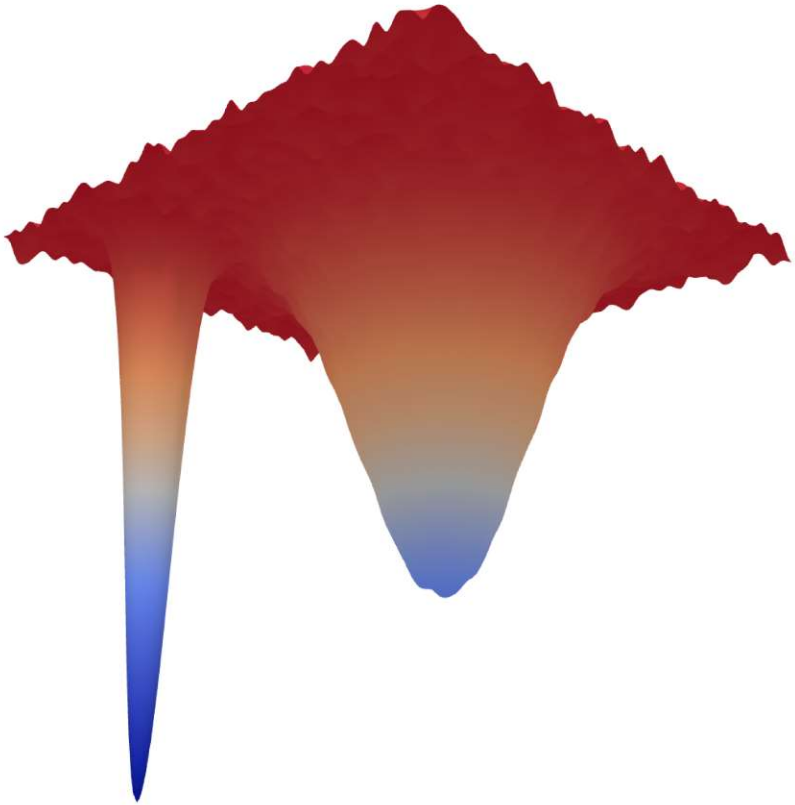
Loss landscape

https://losslandscape.com/explorer

Without regularization



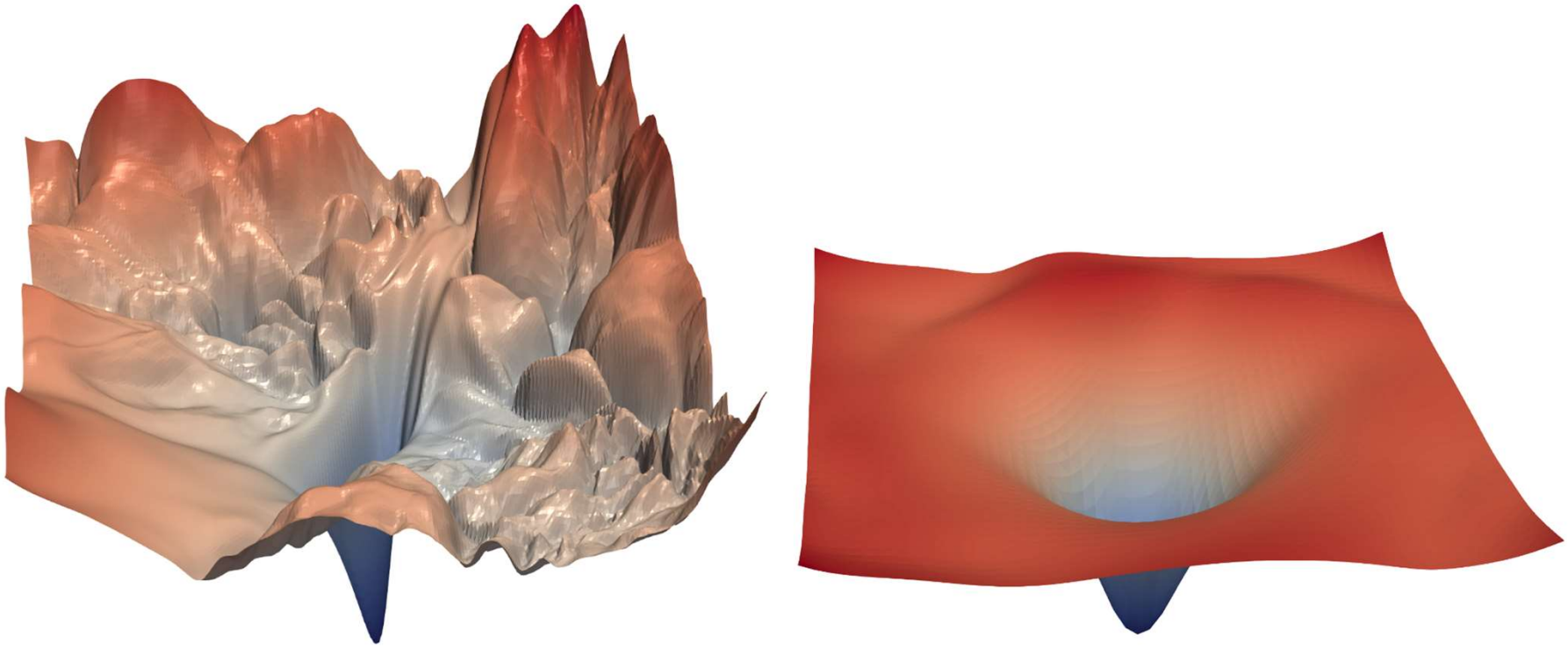
With regularization



Overfitting yields an increased error on unseen data by approximating a simple functional relationship with an overly complex function on the training set

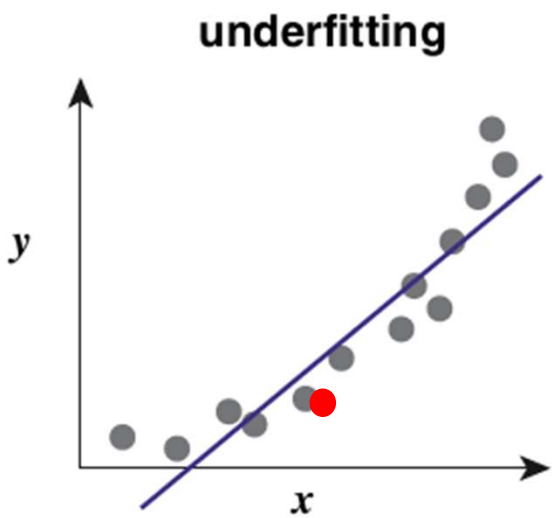
(a) without skip connections

(b) with skip connections



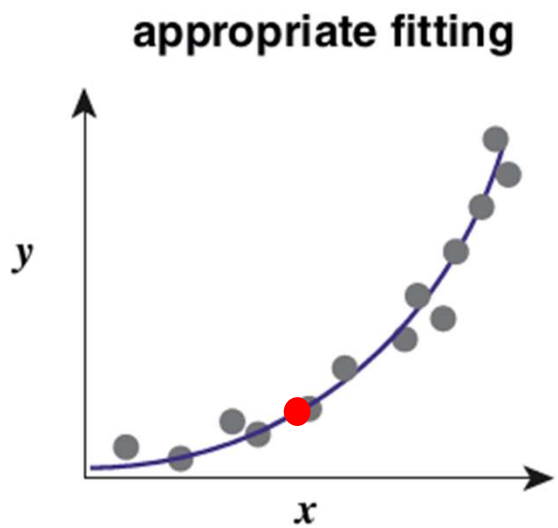
ResNet-56 with and without skip connections

arXiv:1712.09913



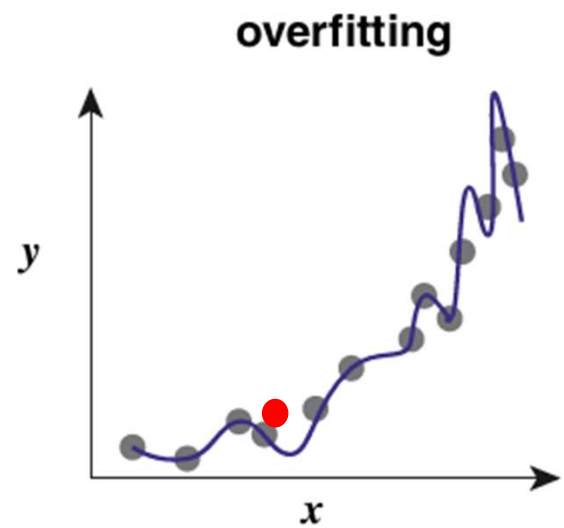
training error: high
validation error: high

Model is not sufficiently complex and flexible to capture data



training error: low
validation error: low

Model is suitably complex to capture trends in data



training error: low
validation error: high

Model is <u>overly</u> complex and will likely not generalize to other samples

How do we quantify the performance of a model?

Residual sum of squares

Coefficient of determination

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - u)^2}$$

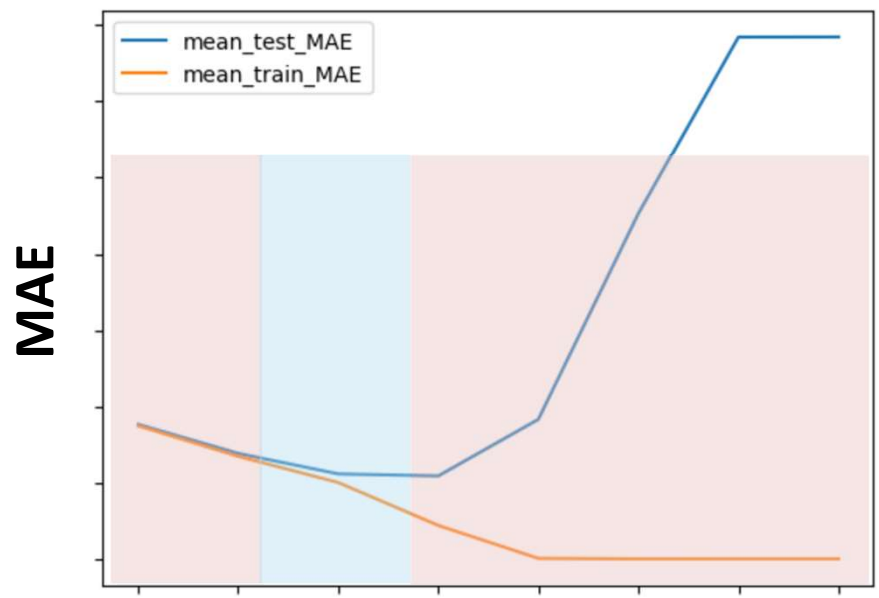
Total sum of squares (distance from mean u)

Mean Absolute Error

$$\mathrm{MAE} = \frac{\sum_i |y_i - f_i|}{n}$$

Root Mean Squared Error

$$\mathrm{RMSE} = \sqrt{\frac{\sum_i (y_i - f_i)^2}{n}}$$

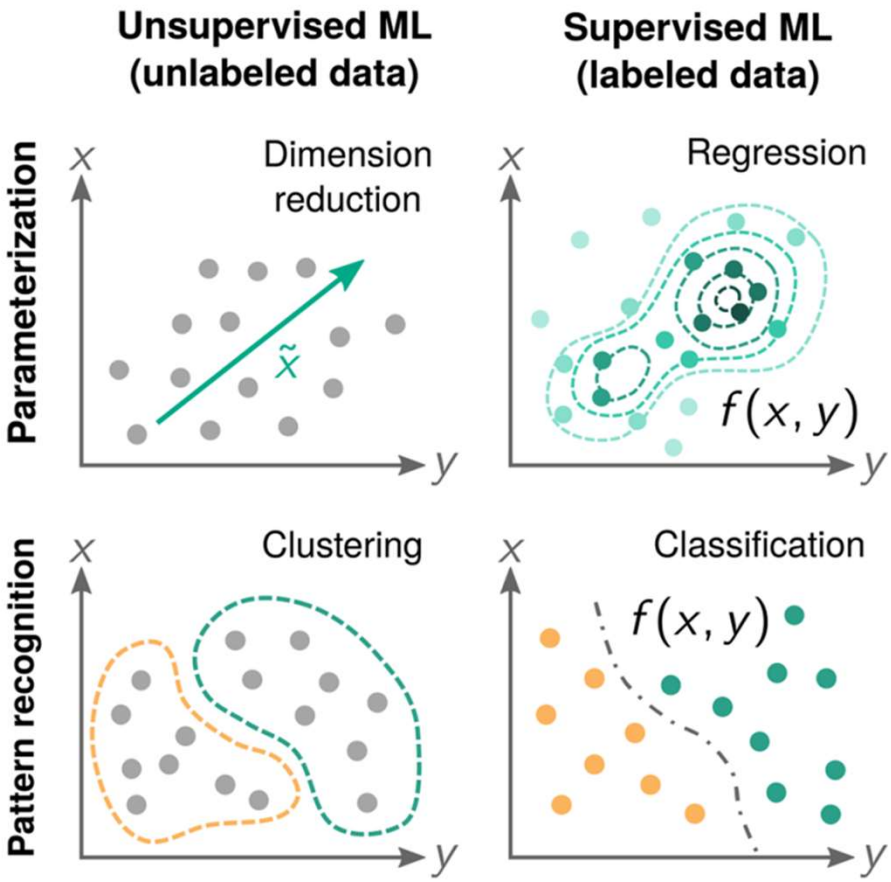


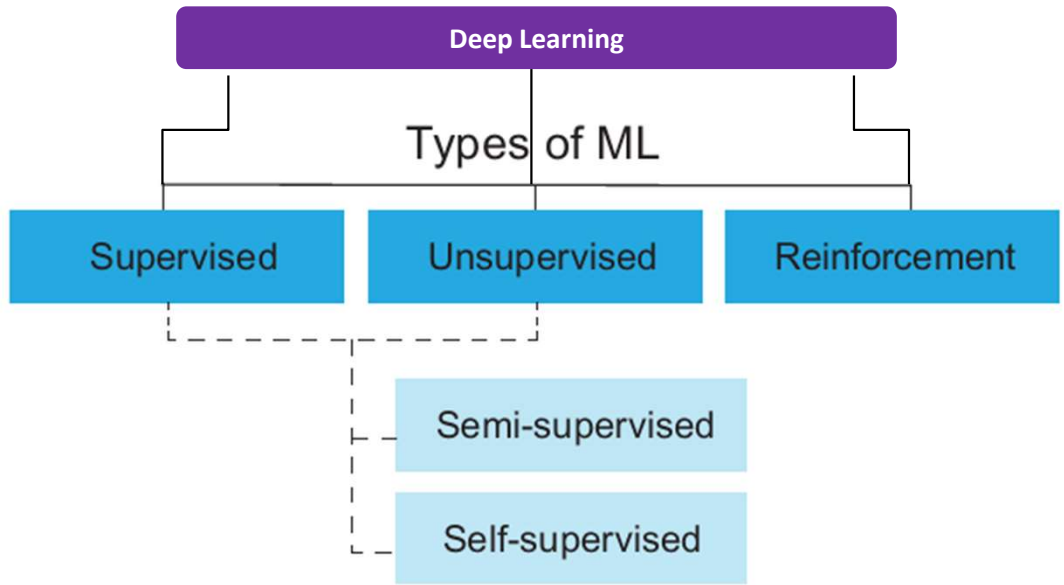
"flexibility of the model"
(e.g. number of parameters)

y_i … ground truth label for data point i
f_i … model prediction for data point i

<u>Try to achieve lowest possible test set error</u>

Types of ML methods





Deep learning approaches exist for all types of ML

Exploratory analysis:
(leading to hypothesis)

- Clustering
- Dimensionality reduction

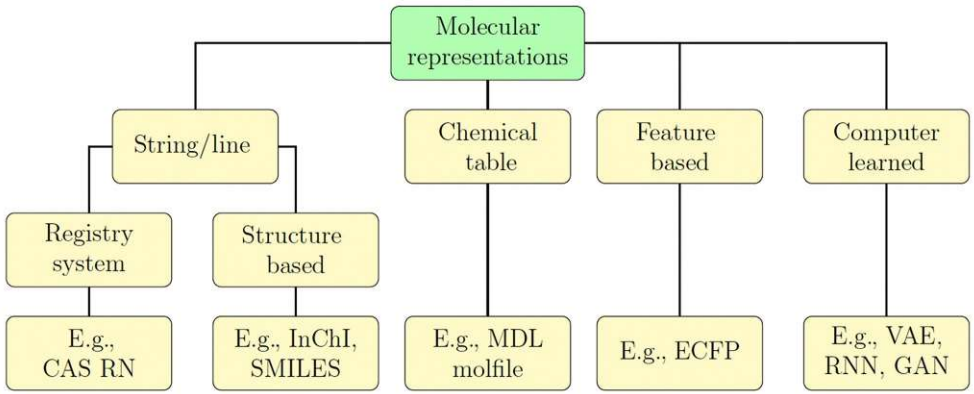
Confirmatory analysis:
(models that test ideas)

- Regression
- Classification

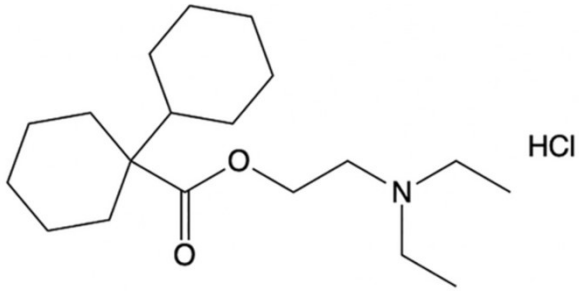
Data Representation and Features

What is the input x in $y = f(x)$?

Examples of Molecular Data Representations



- Lewis structures (2D graph)



- String representations

Generic names[5]	Dicycloverine HCl, benacol, bentyl, dibent, Dyspas, and so on
Mol. formula	$C_{19}H_{36}ClNO_2$
IUPAC name	2-(Diethylamino)ethyl 1-cyclohexylcyclohexane-1-carboxylate hydrochloride
CAS RN	$67-92-5$
Canonical SMILES	CCN(CC)CCOC(=O)C1(CCCCC1)C2CCCCC2.Cl
InChI	InChI = 1S/C19H35NO2.ClH/c1-3-20(4-2)15-16-22-18(21)19
	(13-9-6-10-14-19)17-11-7-5-8-12-17;/h17H,3-16H2,1-2H3;1H
	InChIKey:GUBNMFJOJGDCEL-UHFFFAOYSA-N
WLN[6]	L6TJA-AL6TJAVO2N2&2&GH

- Registry systems

Chemical Abstract Services Registry Number
PubChem CID
ChemSpider
ChEMBL

WIREs Comput Mol Sci. 2022; 12:e1603

SMILES

Simplified Molecular Input Line Entry System
String of ASCII characters that defines a molecule

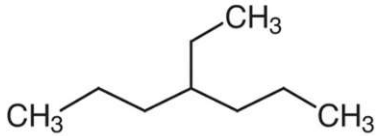
- Atoms are represented with their chemical symbol
- Single bonds are implicit or (-), double bonds (=), triple bonds (#)
- Rings represented via a number after the initial atom and closing atom (e..g C1CCNCC1)
- Branching: represented with parentheses around the branch
- Aromaticity: aromatically bonded atoms in lower case or alternating -C=C-

C1ccccc1,
C1=C-C=C-C=C1,
C1=CC=CC=C1

- Hard to represent unsaturated bonds, radicals, or unusual valences/bonding
- No info on molecular conformation
- Not well suited for generative models

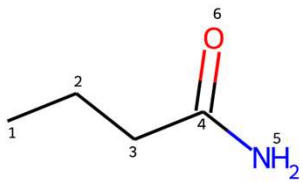
DeepSMILES
SELFIES (self-referencing embedded strings)

4-ethylheptane:
CCCC(CC)CCC



Extended Connectivity Fingerprints, ECFP (a.k.a. Morgan or Circular Fingerprints)

Based on molecular graphs
Interpretable in terms of local atom groups



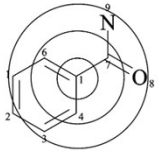
1. Initial assignment stage

```
1: -4080868480043360372
2:  8311098529014133067
3:  8311098529014133067
4: -2155244659601281804
5: -3602994677767288312
6:  8573586092015465947
```

Subgraphs are generated for each atom at each radius and a unique identifier is created with a hash function

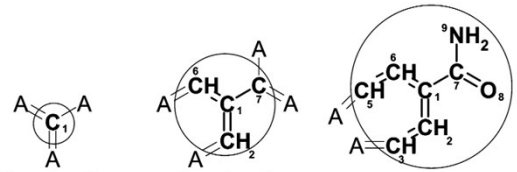
```
import hash from 'crc-32';

const feature = [ 1, 2, 3, 4 ];
const identifier = hash.buf(feature); // -1237517363
```

2. Iterative updating stage (Morgan algorithm



Considering atom 1 in benzoic acid amide

3. Duplicate identifier removal stage

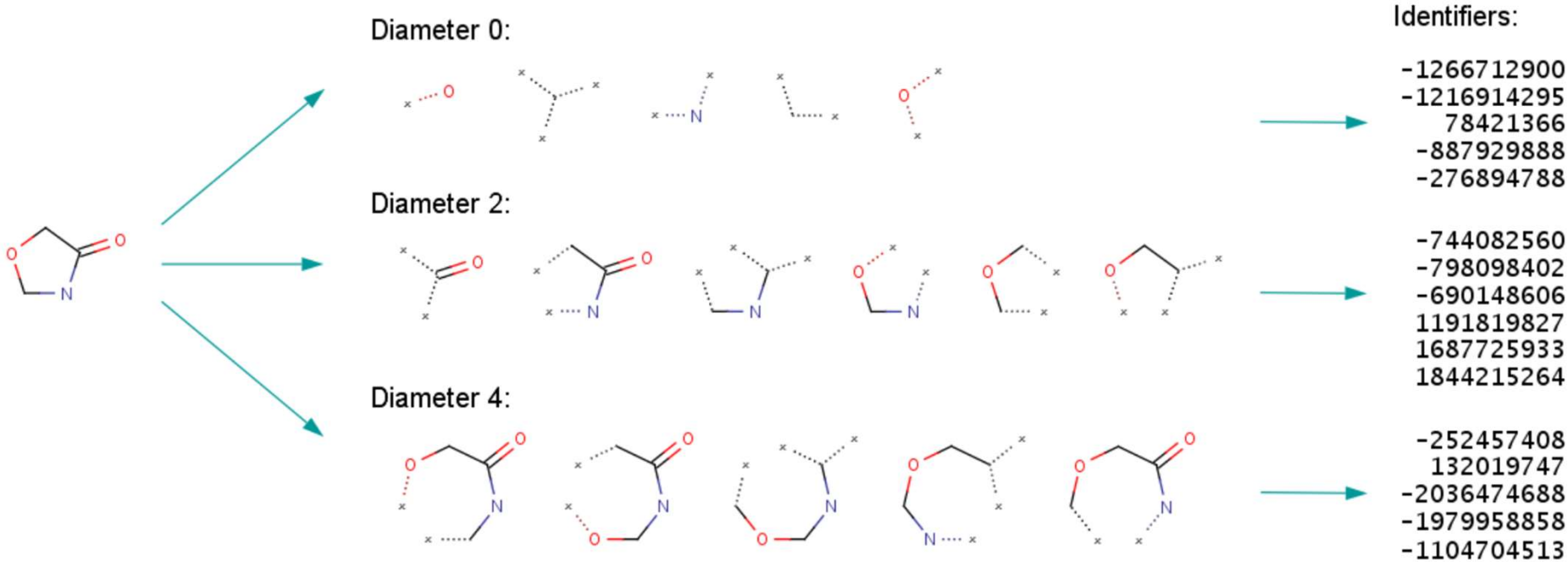


4. Forming the bit array

Implemented in RDKit

Rogers, D.; Hahn, M. "Extended-Connectivity Fingerprints." *J. Chem. Inf. and Model.* **50**:742-54 (2010)

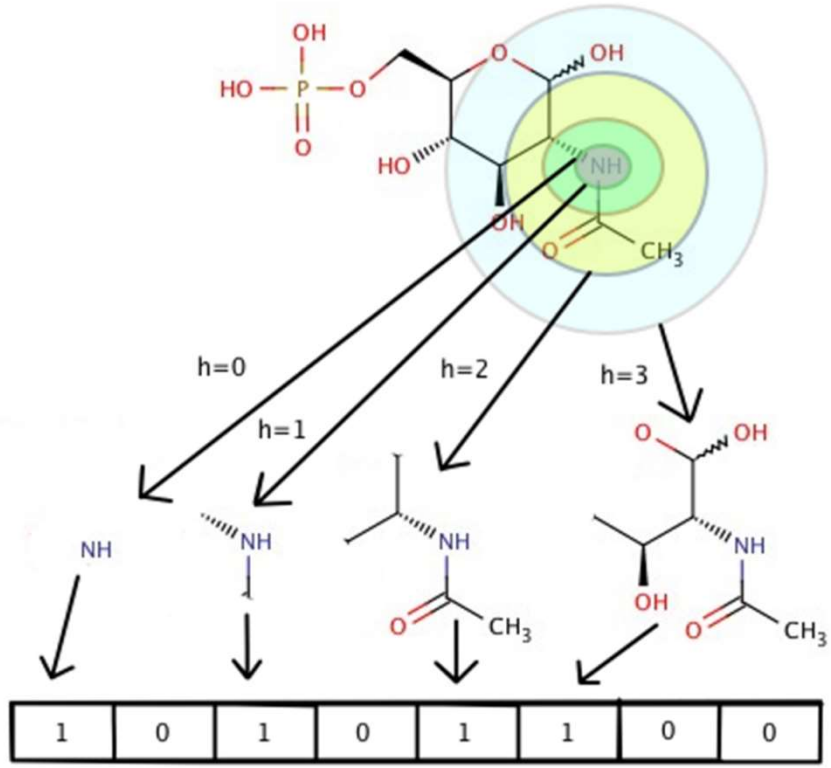
2. Iterative updating stage (Morgan algorithm)



2. Iterative updating stage (Morgan algorithm)

3. Duplicate identifier removal stage

4. Forming the bit array

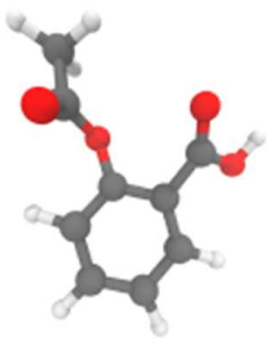


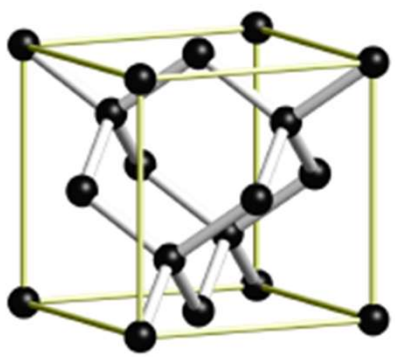
Many more descriptors from cheminformatics
(e.g. SMILES Cliques, molecular fingerprints)
But the typically don't have "atomic resolution"

Data Representation / Featurization for Molecules & Materials

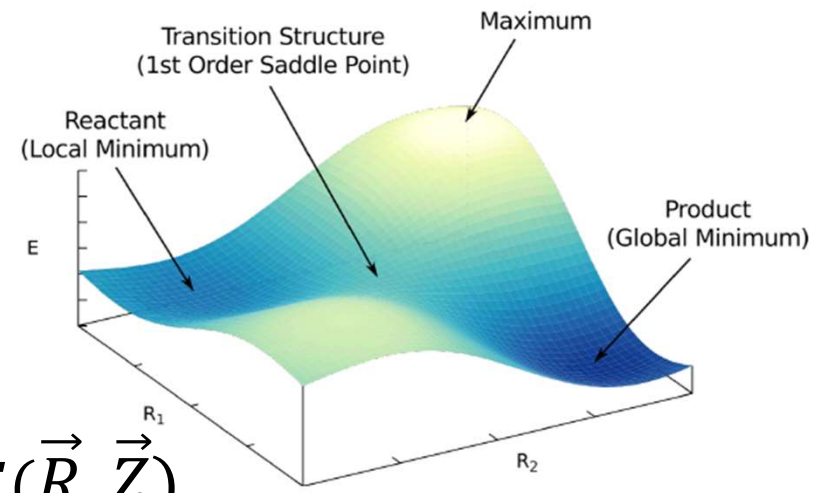
$$\hat{H}\Psi = E\Psi$$

Molecules & Materials





$$S = \{\vec{R}, \vec{Z}\}$$

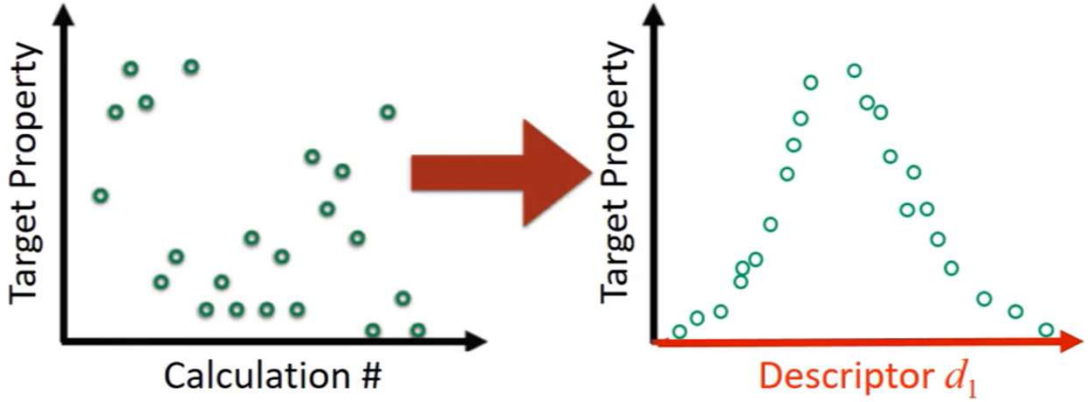


$$E(\vec{R}, \vec{Z})$$

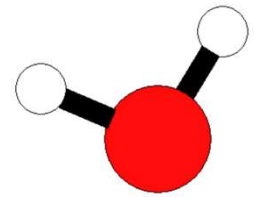
Energy landscape encodes:

- Rotational/Translational invariance
- Permutation invariance
- Symmetry/Local Features

XYZ atomic positions give a terrible representation

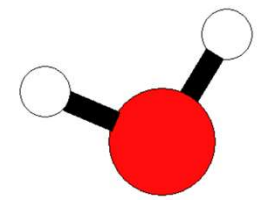


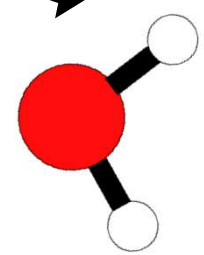
Bad representation for target Good representation



```
[[ 0.        0.        0.119262]
 [ 0.        0.763239 -0.477047]
 [ 0.       -0.763239 -0.477047]]
```

```
[[5.        7.        9.119262]
 [5.        7.763239  8.522953]
 [5.        6.236761  8.522953]]
```

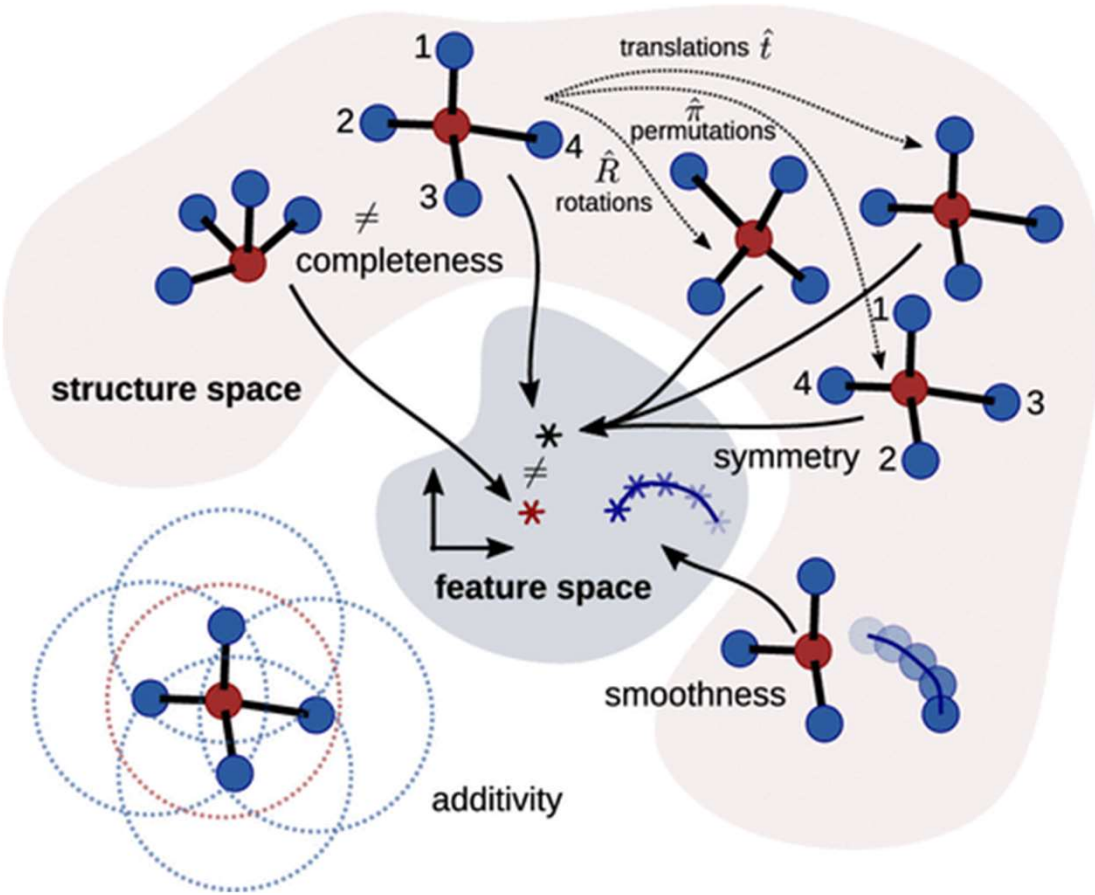




XYZ coordinates do not capture the basic structural symmetry properties of molecules and materials

```
[[-7.        5.        9.119262]
 [-7.763239  5.        8.522953]
 [-6.236761  5.        8.522953]]
```

Requirements on molecular representations (a.k.a. descriptors)



Chem. Rev. 2021, 121, 16, 9759–9815

Symmetry Invariance:
Mapping onto the same point in feature space

Completeness & Uniqueness:
Being able to distinguish all symmetry-inequivalent arrangements by mapping onto different points in feature space

Smoothness:
Smooth changes in atomic positions lead to smooth changes in feature space

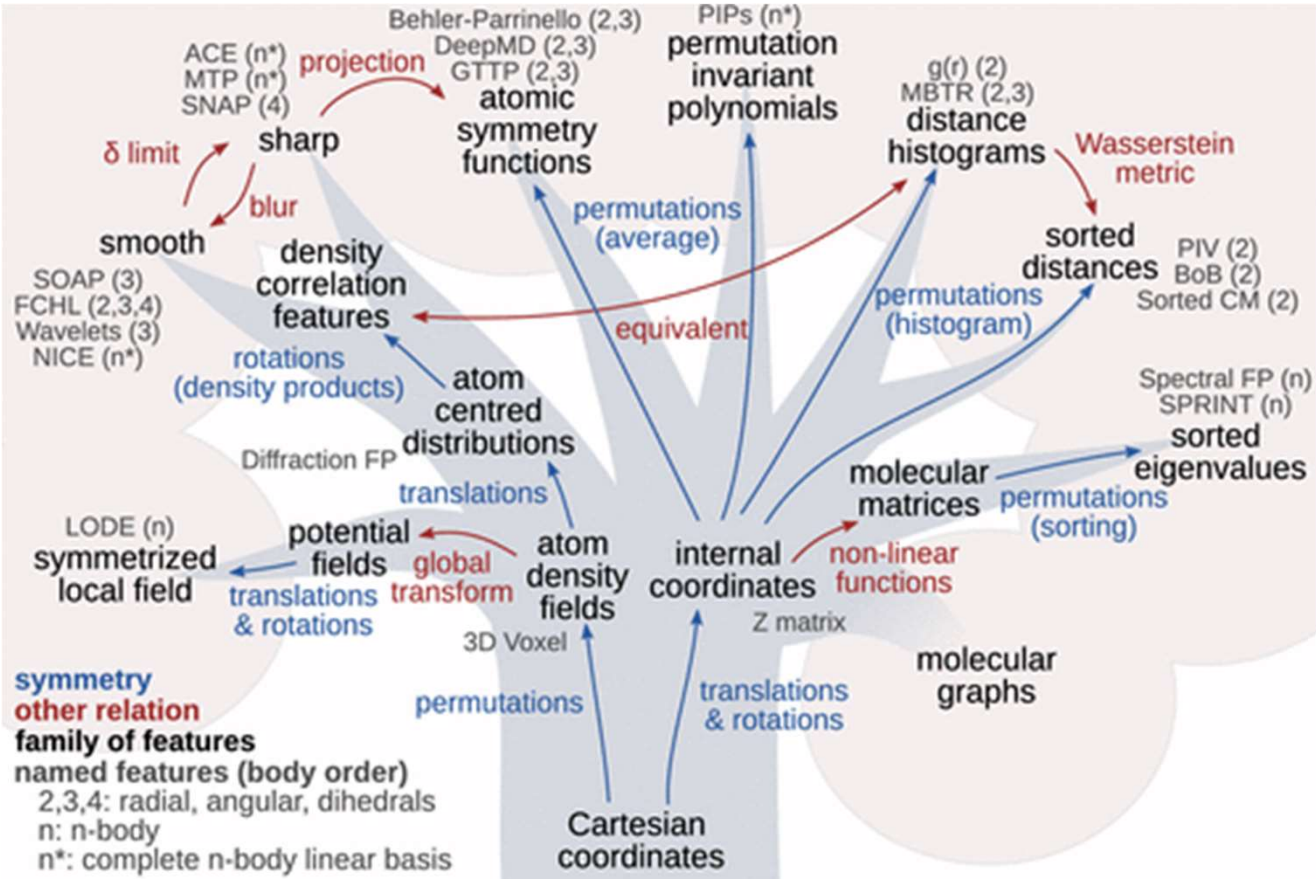
Nomenclature

- **Structure:** Atomic positions, chemical species, unit cell
- **Descriptor:** A specific method for transforming the structure of different molecules/materials into a constant vector with correct symmetry properties
- **Feature vector:** A vector of numbers produced by converting the structure according to a certain descriptor. The feature vector is the structural representation in “feature space”

Global descriptors

Atom-wise descriptors

There are many atomic descriptors …



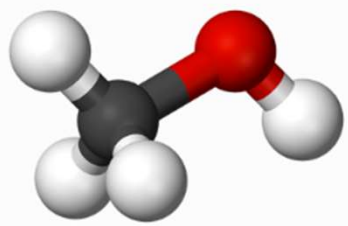
I will cover

- Coulomb Matrix (CM)
- MBTR
- SOAP
- ACE

Chem. Rev. 2021, 121, 16, 9759–9815

Coulomb Matrix

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



$$\begin{bmatrix} 36.9 & 33.7 & 5.5 & 3.1 & 5.5 & 5.5 \\ 33.7 & 73.5 & 4.0 & 8.2 & 3.8 & 3.8 \\ 5.5 & 4.0 & 0.5 & 0.35 & 0.56 & 0.56 \\ 3.1 & 8.2 & 0.35 & 0.5 & 0.43 & 0.43 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.5 & 0.56 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.56 & 0.5 \end{bmatrix}$$

- A **global** representation
- Nuclear charges and atomic distances
- Doesn't work for periodic systems
- Molecules of different sizes require padding with zeros
- Adaptations for periodic systems exist

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

Many Body Tensor Representation (MBTR)

- For each pair, triple, etc. of atom types, MBTR encodes a “spectrum of distances”

$$f_2(x, z_1, z_2) = \sum_{i,j=1}^{N_a} w_2(i,j)\, \mathcal{D}(x, g_2(i,j))\, C_{z_1,Z_i} C_{z_2,Z_j}$$

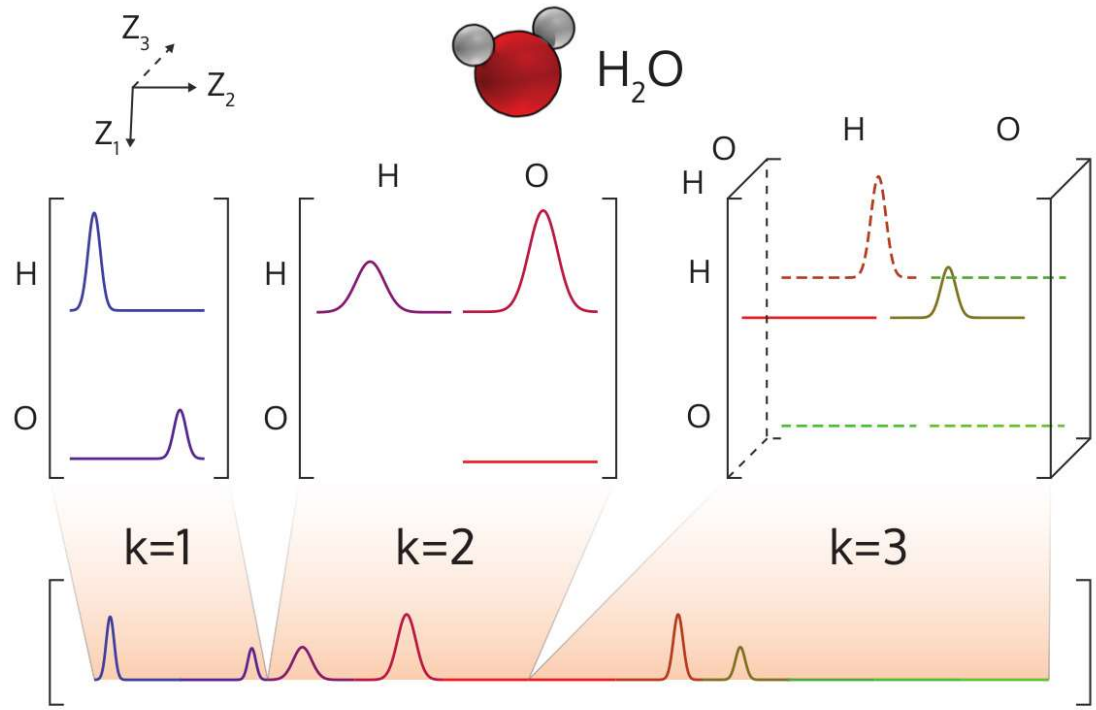
$$f_k(x, \boldsymbol{z}) = \sum_{\boldsymbol{i}=1}^{N_a} w_k(\boldsymbol{i})\, \mathcal{D}(x, g_k(\boldsymbol{i})) \prod_{j=1}^{k} C_{z_j,Z_{i_j}}$$

$g_2(i,j)$ … inverse distances

$g_3(i,j)$ … “angle distance metric”

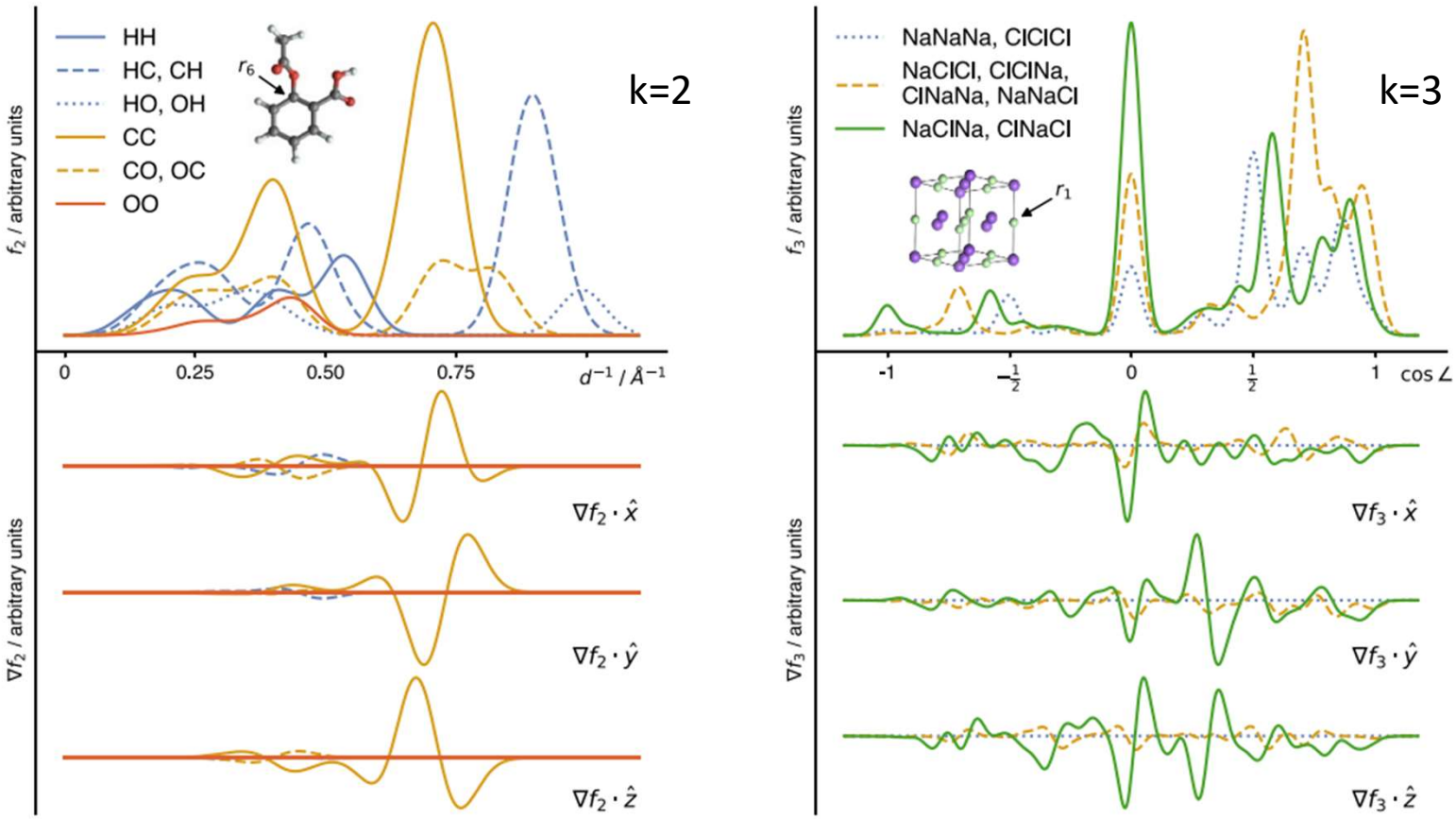
w_k … weighting

$D(x, g_k)$ … broadening



Haoyan Huo and Matthias Rupp 2022 *Mach. Learn.: Sci. Technol.* **3** 045017

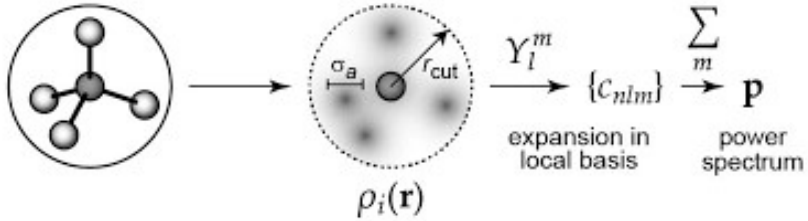
Many Body Tensor Representation (MBTR)



differentiable, continuous, fast, versatile, interpretable

Smooth Overlap of Atomic Positions (SOAP)

- Smooth Overlap of Atomic Positions (SOAP) encodes regions of atomic geometries ("atomic environments")
- One set of features for each atom in the system with a cutoff
- Satisfies all relevant symmetries and feature requirements
- Perfect for interatomic potentials, but can be slow to calculate



Density smearing with Gaussians

$$\rho(\mathbf{r}) = \sum_i \exp\left(-\alpha\|\mathbf{r}-\mathbf{r}_i\|_2^2\right) f_{\text{cut}}(|\mathbf{r}_i|)$$

Representation via radial funcs and spherical harmonics

$$\rho(\mathbf{r}) = \sum_{nlm} c_{nlm} R_n(r) Y_{lm}(\hat{\mathbf{r}})$$

Distance metric for each component of SOAP vector by averaging over rotations

$$k(\rho,\rho') = \int \left|S\left(\rho,\hat{R}\rho'\right)\right|^2 \mathrm{d}\hat{R} = \sum_{l,m,m'} \left(I^l_{mm'}\right)^* I^l_{mm'}$$

Products of Clebsch-Gordan coeffs.

$$I^l_{mm'} = \sum_n c_{nlm}\left(c'_{nlm'}\right)^* \qquad p_{nn'l} = \sum_m c^*_{nlm} c_{n'lm}$$

$$k(\rho,\rho') = \sum_{n,n',l,m,m'} \overbrace{c_{nlm}\left(c'_{nlm'}\right)^*}\left(c_{nlm}\right)^* c'_{n'lm'}$$

Atomic Cluster Expansion (ACE)

Designed for interatomic potential construction

$$E = \sum_i \varepsilon_i$$

Atomic energy contributions

Many-body cluster expansion

$$\varepsilon_i = V^{(0)}(Z_i) + \sum_{j_1} V^{(1)}(\boldsymbol{x}_{ij_1}) + \sum_{j_1<j_2} V^{(2)}(\boldsymbol{x}_{ij_1}, \boldsymbol{x}_{ij_2}) + \cdots + \sum_{j_1<\cdots<j_{\bar{\nu}}} V^{(\bar{\nu})}(\boldsymbol{x}_{ij_1}, \ldots, \boldsymbol{x}_{ij_{\bar{\nu}}}) \quad (2a)$$

Expand each body order component in a basis

$$V^{(1)}(\boldsymbol{x}_{ij_1}) = \sum_{k_1} c_{k_1}^{(Z_i)} \phi_{k_1}(\boldsymbol{x}_{ij_1})$$

$$V^{(2)}(\boldsymbol{x}_{ij_1}, \boldsymbol{x}_{ij_2}) = \sum_{k_1,k_2} c_{k_1k_2}^{(Z_i)} \phi_{k_1}(\boldsymbol{x}_{ij_1}) \phi_{k_2}(\boldsymbol{x}_{ij_2})$$

$$\vdots \qquad \vdots$$

$$V^{(\bar{\nu})}(\boldsymbol{x}_{ij_1}, \ldots, \boldsymbol{x}_{ij_{\bar{\nu}}}) = \sum_{k_1,\ldots,k_{\bar{\nu}}} c_{k_1\ldots k_{\bar{\nu}}}^{(Z_i)} \phi_{k_1}(\boldsymbol{x}_{ij_1}) \cdots \phi_{k_{\bar{\nu}}}(\boldsymbol{x}_{ij_{\bar{\nu}}})$$

Atomic basis: radial functions and spherical harmonics

$$\phi_{znlm}(\boldsymbol{r}_{ij}, Z_i, Z_j) = R_{nl}(r_{ij}, Z_i, Z_j) Y_l^m(\hat{\boldsymbol{r}}_{ij}) \delta_{zZ_j}$$

$$A_{znlm}^i = \sum_{j \in \mathcal{N}(i)} \phi_{znml}(\boldsymbol{r}_{ij}, Z_j, Z_i)$$

where $\mathcal{N}(i)$ denotes the set of indices of all atoms within the cutoff radius from atom i.

Body/correlation order products of basis functions

Product basis: for lexicographically ordered tuples $(\boldsymbol{z}, \boldsymbol{n}, \boldsymbol{l}, \boldsymbol{m}) = (z_t, n_t, l_t, m_t)_{t=1}^{\nu}$ we define

$$\boldsymbol{A}_{\boldsymbol{znlm}}^i = \prod_{t=1}^{\nu} A_{z_t n_t l_t m_t}^i. \quad (A2)$$

Drautz, Phys. Rev. B **99**, 014104 (2019)

Atomic Cluster Expansion (ACE)

Making basis functions rotationally invariant

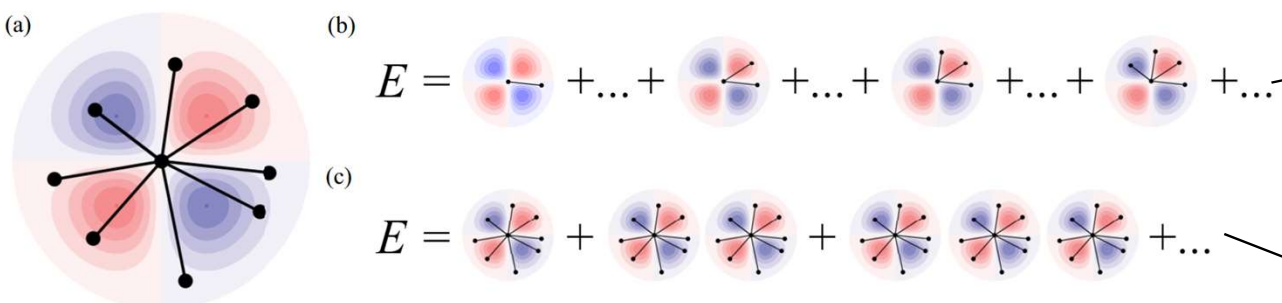
$$B^{(1)}_{in} = A_{in00},$$

$$B^{(2)}_{in_1n_2l} = \sum_{m=-l}^{l} (-1)^m A_{in_1lm} A_{in_2l-m},$$

$$B^{(3)}_{\substack{in_1n_2n_3\\ l_1l_2l_3}} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} \sum_{m_3=-l_3}^{l_3} \begin{pmatrix} l_1 & l_2 & l_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \times A_{in_1l_1m_1} A_{in_2l_2m_2} A_{in_3l_3m_3},$$

$$\varepsilon_i = \sum_n c^{(1)}_n B^{(1)}_{in} + \sum_{n_1n_2l}{}' c^{(2)}_{n_1n_2l} B^{(2)}_{in_1n_2l} + \sum_{\substack{n_1n_2n_3\\ l_1l_2l_3}}{}' c^{(3)}_{\substack{n_1n_2n_3\\ l_1l_2l_3}} B^{(3)}_{\substack{in_1n_2n_3\\ l_1l_2l_3}}$$

$$\varepsilon_i = \sum_{K\boldsymbol{nl}} \boldsymbol{c}^{(K)}_{\boldsymbol{nl}} \boldsymbol{B}^{(K)}_{\boldsymbol{inl}}$$



$$E_i(\boldsymbol{\sigma}) = \sum_j \sum_v c^{(1)}_v \phi_v(\boldsymbol{r}_{ji}) + \frac{1}{2} \sum_{j_1j_2} \sum_{v_1v_2} c^{(2)}_{v_1v_2} \phi_{v_1}(\boldsymbol{r}_{j_1i}) \phi_{v_2}(\boldsymbol{r}_{j_2i}) + \frac{1}{3!} \sum_{j_1j_2j_3} \sum_{v_1v_2v_3} c^{(3)}_{v_1v_2v_3} \phi_{v_1}(\boldsymbol{r}_{j_1i}) \phi_{v_2}(\boldsymbol{r}_{j_2i}) \phi_{v_3}(\boldsymbol{r}_{j_3i})$$

$$E_i(\boldsymbol{\sigma}) = \sum_v c^{(1)}_v A_{iv} + \sum_{v_1v_2}^{v_1 \geqslant v_2} c^{(2)}_{v_1v_2} A_{iv_1} A_{iv_2} + \sum_{v_1v_2v_3}^{v_1 \geqslant v_2 \geqslant v_3} c^{(3)}_{v_1v_2v_3} A_{iv_1} A_{iv_2} A_{iv_3} + \cdots$$

$$A^i_{znlm} = \sum_{j \in \mathcal{N}(i)} \phi_{znml}(\boldsymbol{r}_{ij}, Z_j, Z_i)$$

ACE is a generalization of many possible atom-centred descriptors

ACE connects to SOAP, ACSFs (Behler), SOAPs, Steinhardt parameters

Distance metric for each component of SOAP vector by averaging over rotations

$$k(\rho, \rho') = \sum_{n,n',l,m,m'} c_{nlm} \left(c'_{nlm'}\right)^* \left(c_{nlm}\right)^* c'_{n'lm'}.$$

ACE correlation order 2 terms

$$k(\varrho, \varrho') = \sum_{n_1 n_2 l} B^{(2)}_{n_1 n_2 l}(\varrho) B^{(2)}_{n_1 n_2 l}(\varrho').$$

$$B^{(2)}_{i n_1 n_2 l} = \sum_{m=-l}^{l} (-1)^m A_{i n_1 l m} A_{i n_2 l -m}$$

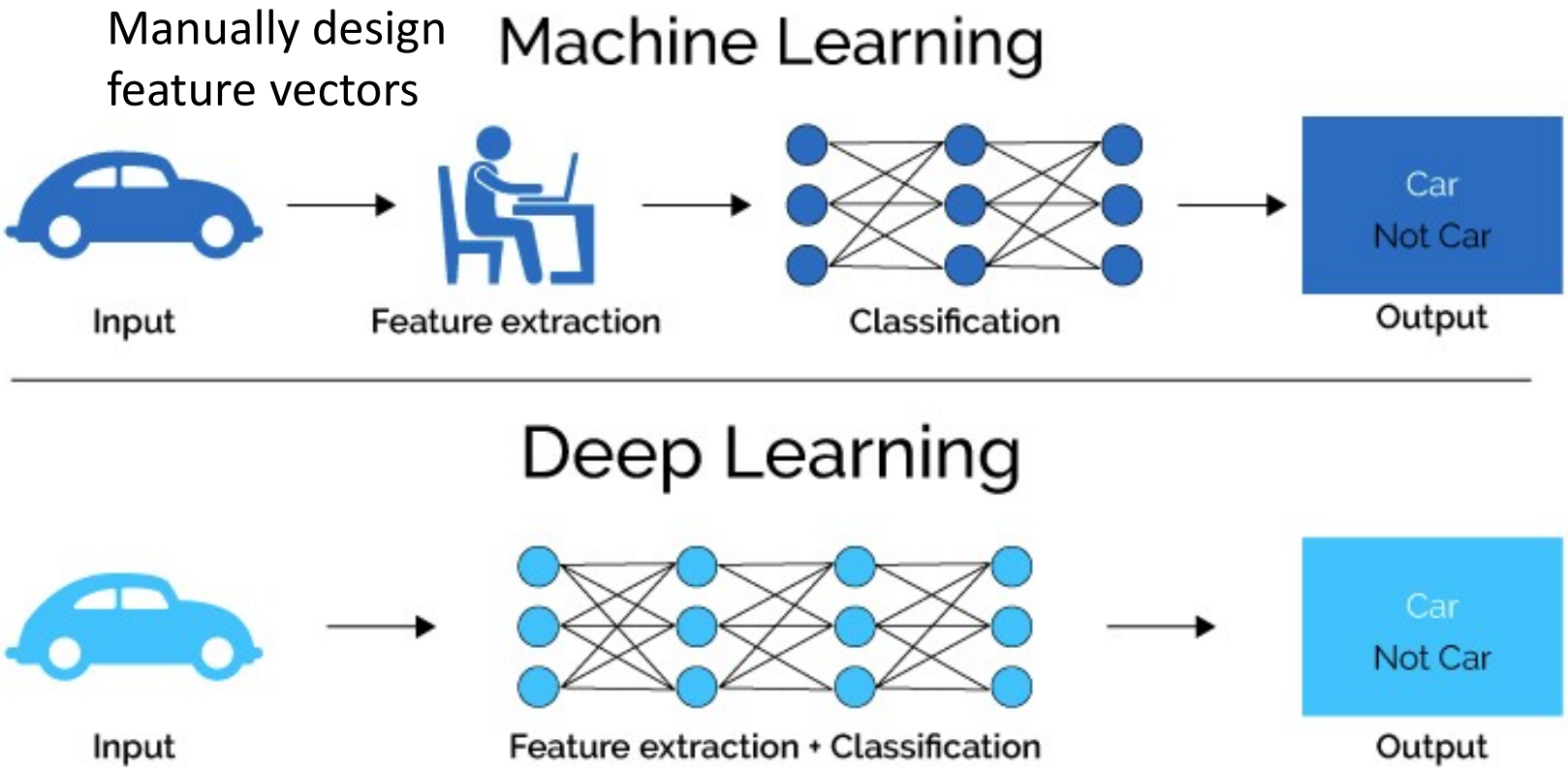
$$A^i_{znlm} = \sum_{j \in \mathcal{N}(i)} \phi_{znml}(\boldsymbol{r}_{ij}, Z_j, Z_i)$$

$$\phi_{znlm}(\boldsymbol{r}_{ij}, Z_i, Z_j) = R_{nl}(r_{ij}, Z_i, Z_j) Y^m_l(\hat{\boldsymbol{r}}_{ij}) \delta_{zZ_j}$$

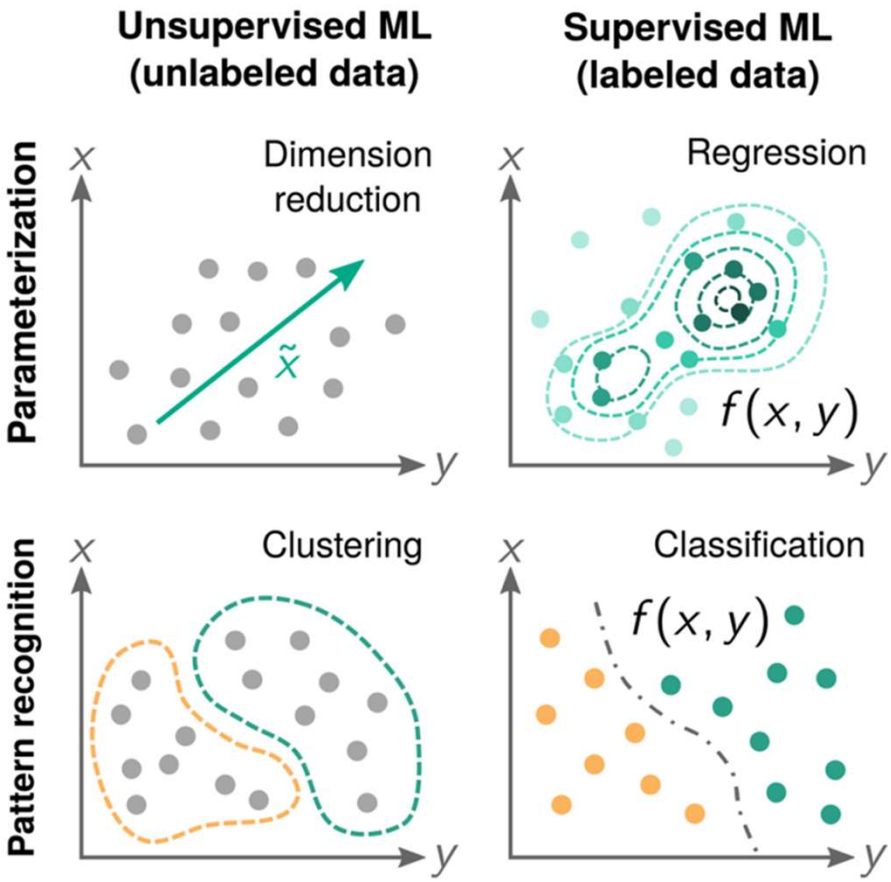
SOAP is similar to ACE with body order 3 (correlation order 2)

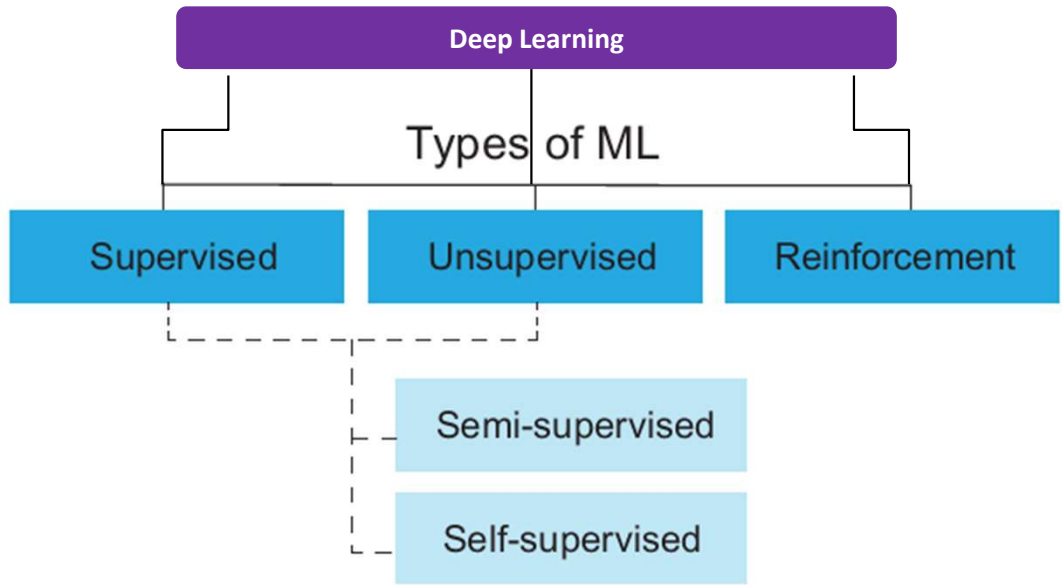
Side note on features

Manually design feature vectors



Types of ML methods





Deep learning approaches exist for all types of ML

Exploratory analysis:
(leading to hypothesis)

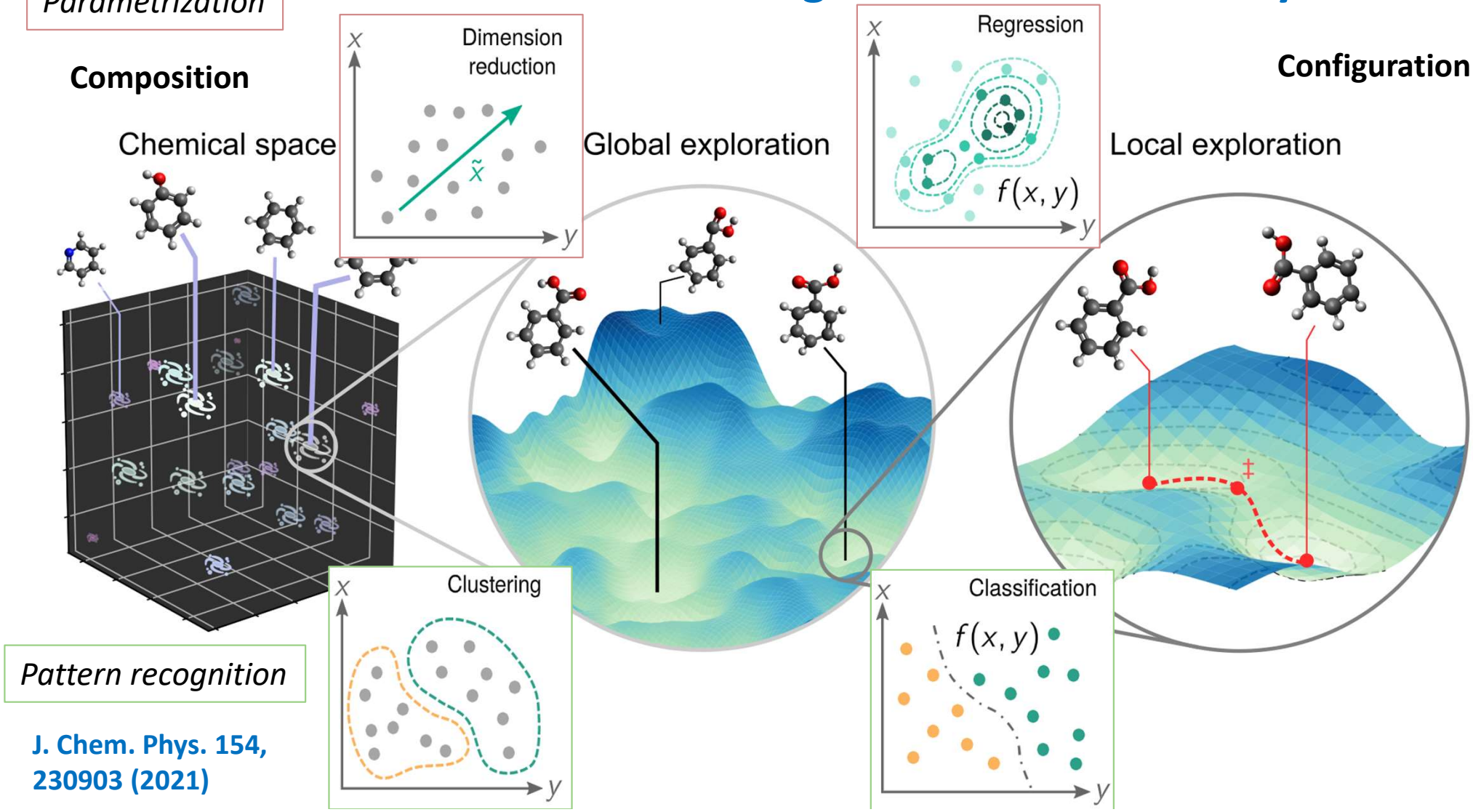
- Clustering
- Dimensionality reduction

Confirmatory analysis:
(models that test ideas)

- Regression
- Classification

ML for chemical space exploration

Machine learning methods in chemistry

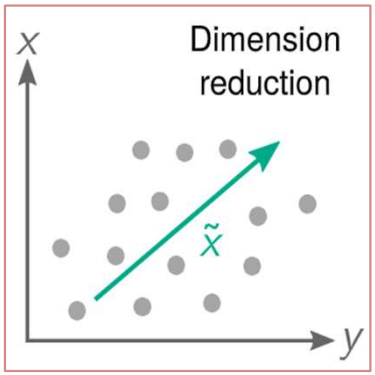


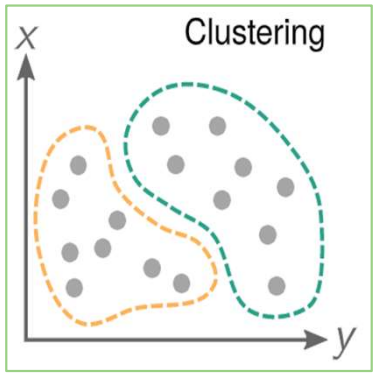
J. Chem. Phys. 154, 230903 (2021)

Unsupervised Machine Learning

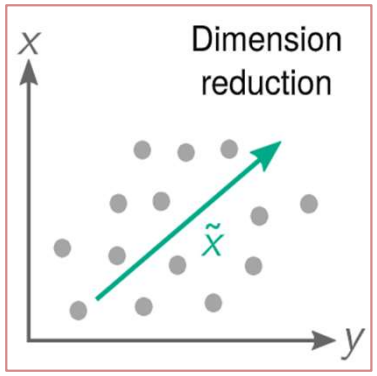
Constructing a model from input data without corresponding output labels
Goal: describe or understand the structure of the data.

- Dimensionality Reduction
- Clustering
- Outlier detection
- Generative Machine Learning





Unsupervised ML: Dimensionality Reduction



Dimensionality Reduction is crucial for identifying the smallest number of features that contain as much information as possible

Principal Component Analysis (PCA)

Start with input vectors $\boldsymbol{x}$ and build covariance matrix $\boldsymbol{Q} = \boldsymbol{x}^T \boldsymbol{x}$

Covariance matrix tells us how similar each of the inputs x_i are (all pairwise dot products of x_i)

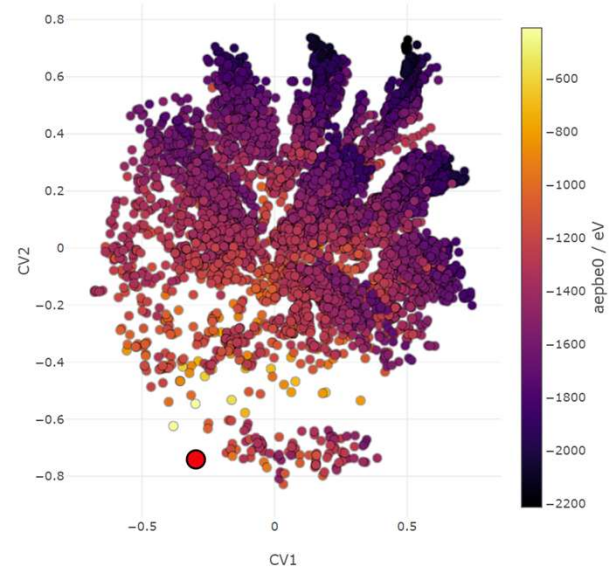
Diagonalize $Q = \boldsymbol{W}\,\boldsymbol{\Lambda}\,\boldsymbol{W}^T$

Principal Component eigenvalues $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$

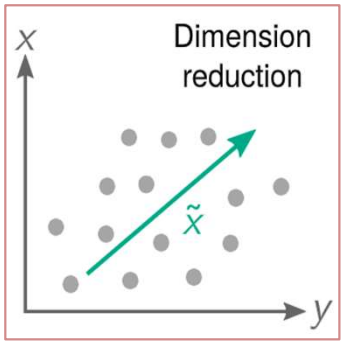
Biggest eigenvalues are responsible for most of the data variance

Principal Component eigenvectors W tell us how inputs $\boldsymbol{x_i}$ mix

Select a subset of L principal components and transform into lower dimensional space $\tilde{\boldsymbol{X}}_L = \boldsymbol{X}\boldsymbol{W}_L$



Unsupervised ML: Dimensionality Reduction



Dimensionality Reduction can help to recognize and visualize patterns in data

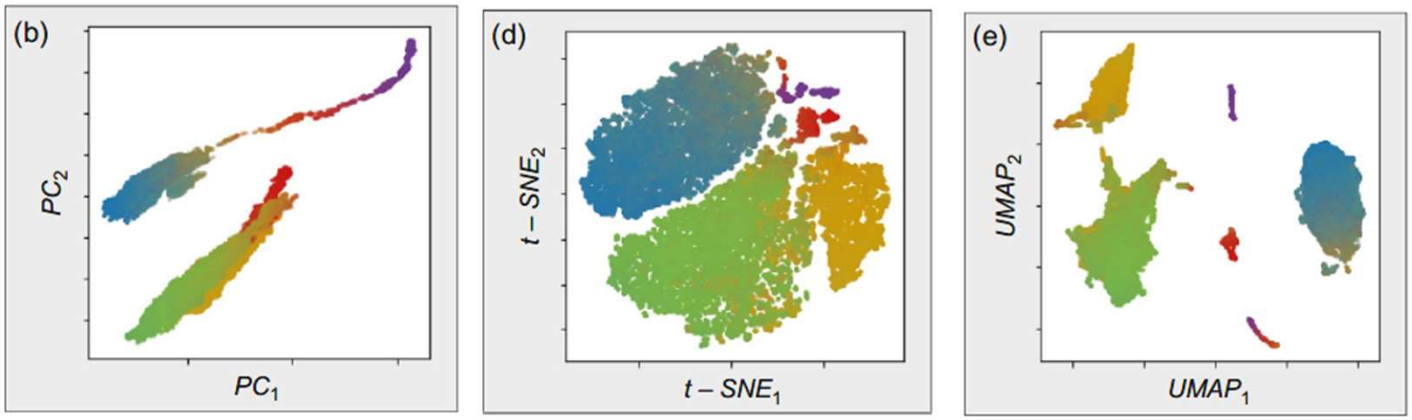
T-SNE: t-distributed stochastic neighbor embedding

t-SNE focuses on preserving the pairwise similarities between data points in a lower-dimensional space

Both are non-linear mappings

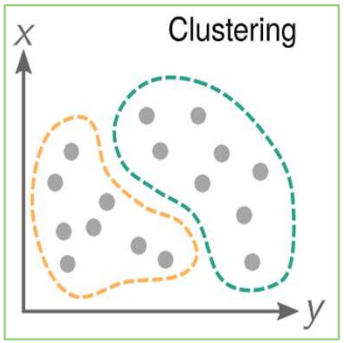
UMAP: Uniform Manifold Approximation and Projection

Similar to t-SNE but uses tricks of topological data analyses to reduce comp. overhead



t-SNE preserves small pairwise similarities whereas,
PCA maintains large pairwise distances to maximize variance.

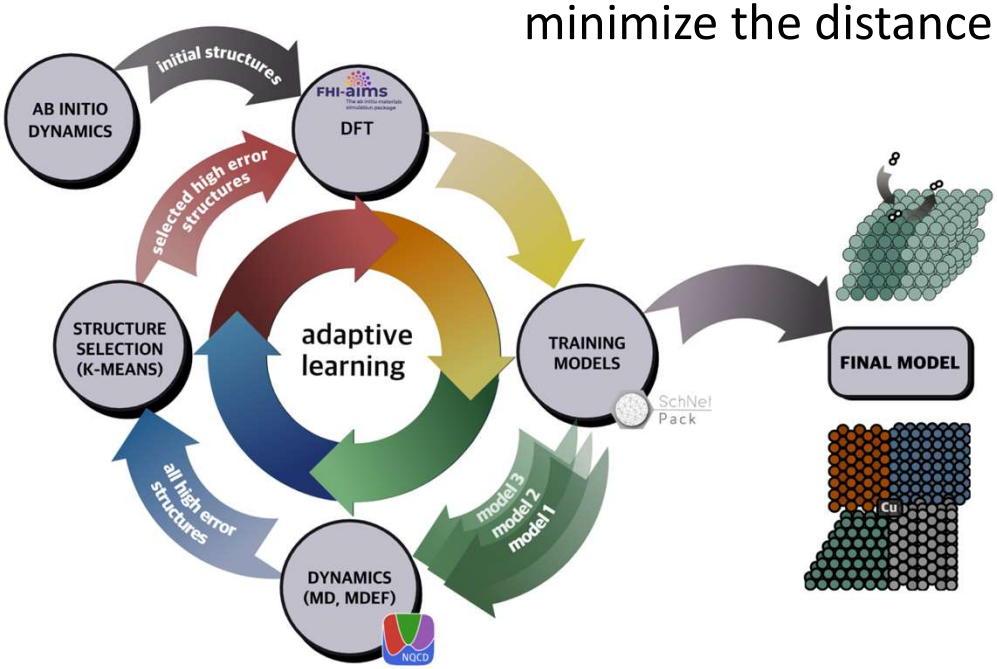
Unsupervised ML: Clustering

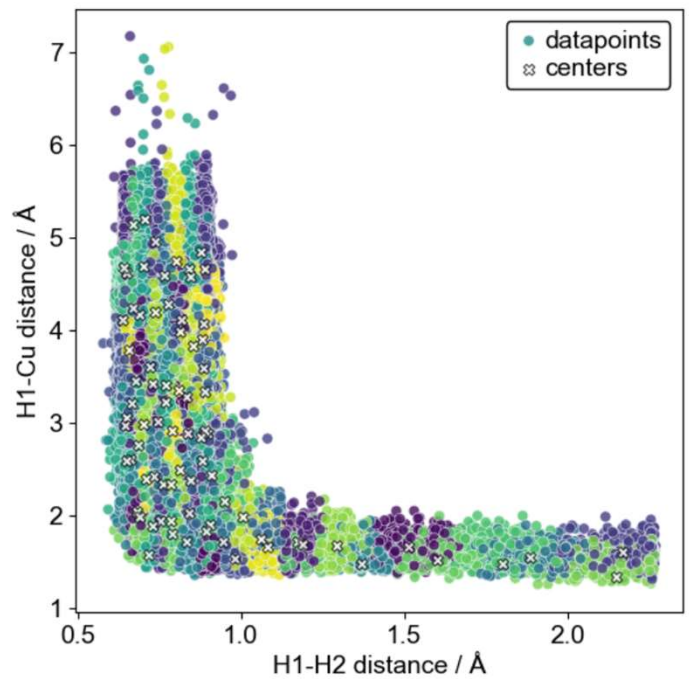


Example: **K-Means** algorithm

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

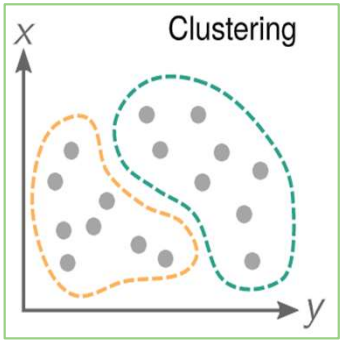
- Clusters n data points by separating them into k clusters of equal variance
- Requires number of clusters as input
- The algorithm chooses cluster centroids that minimize the distance to each point in the cluster





Stark et al. J. Phys. Chem. C 127, 24168-24182

Unsupervised ML: Clustering



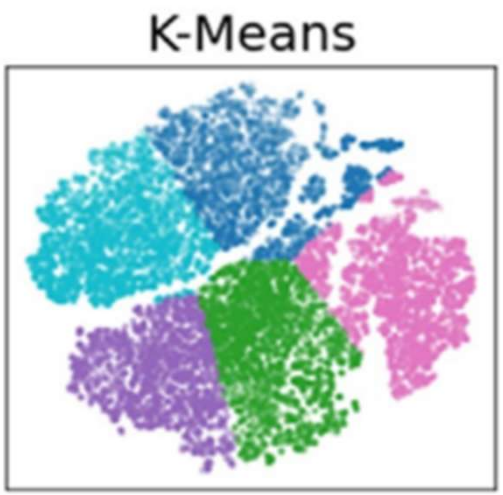
Partition-based clustering vs. Density-based clustering

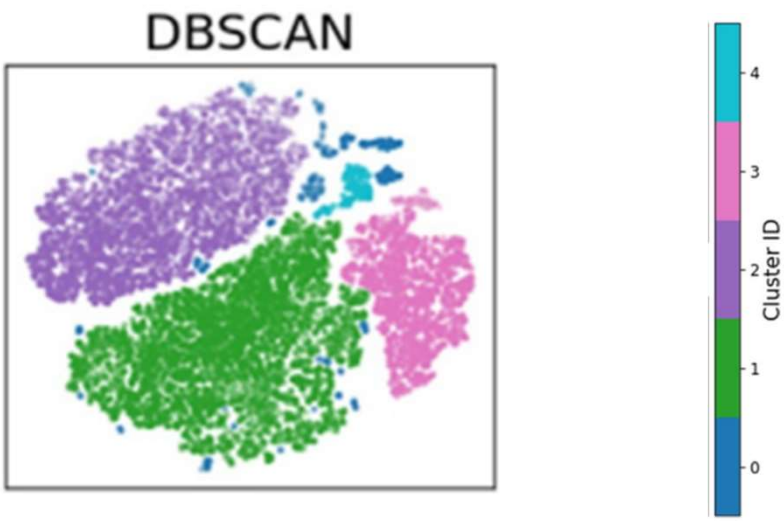
Find clusters of equal size

e.g. **K-Means**

Find variable size clusters around regions of high density

e.g. **DBSCAN, HDBSCAN**



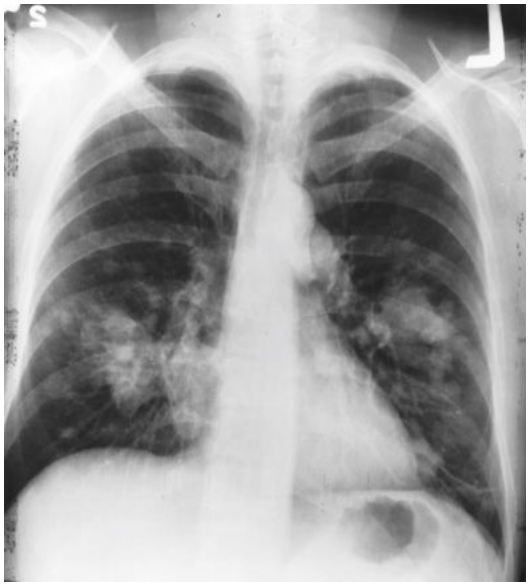


Semi-supervised learning

- A training dataset with both – labeled and unlabeled – data
- when extracting relevant features from data is difficult
- labeling examples is a time-intensive task for experts

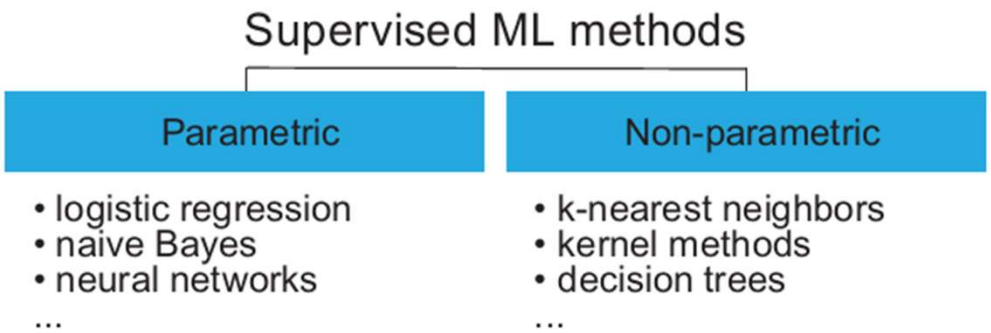
Examples:

- Medical images like MRI images
- GANs: Generative adversarial networks
  - Generator (generates output)
  - Discriminator (critiques output)
  - Battling against each other
  - Network itself provides labels



Supervised Machine Learning

Create model: $y = f(x)$



Discrete output space (classification)
Continuous output space (regression)

<u>Parametric model:</u>
Number of model parameters are independent of number of training datapoints. Model has a fixed size

<u>Non-parametric model:</u>
Number of model parameters depend on the number of training data points.

Models we will discuss:

- Multivariate Linear Regression
- Kernel Ridge Regression
- Decision Trees

Example: pKa prediction of substituted phenols based on pKa of related benzoic acid

$$f(x_i) = y_i$$

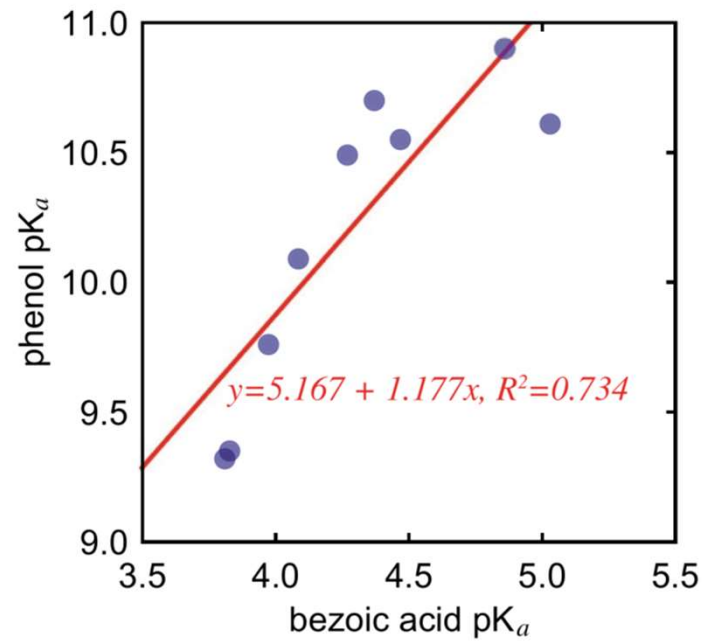
inputs (features)

output

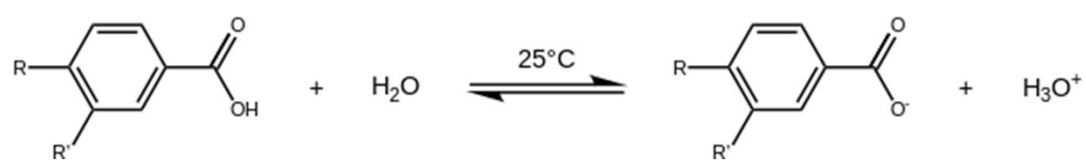
TABLE 1 pK_a of some substituted benzoic acids in water at 25°C [3], and the corresponding substituted phenols in 20% water-ethanol (v/v) at 20°C [4].

Substituent	Substituted benzoic acids	Substituted phenols
H	4.200	10.30
m-NH_2	4.360	10.37
p-NH_2	4.860	10.90
p-$N(CH_3)_2$	5.030	10.61
m-OCH_3	4.085	10.09
p-OCH_3	4.468	10.55
m-Br	3.809	9.32
m-Cl	3.827	9.35
p-Cl	3.973	9.76
m-CH_3	4.269	10.49

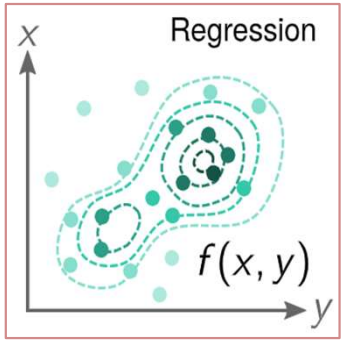
Training data: 10 data points



Hammett equation



Multivariate Linear Regression

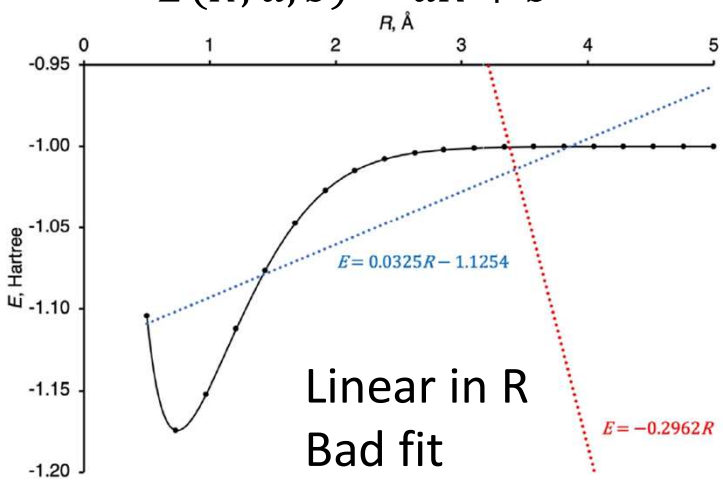


- Linear fit in high dimensional space
- Find a set of regression coefficients β
- Important that input data x_i is represented in a way that shows close to linear relationship with y_i
- Many ways to fit a large set of parameters
- Not a universal estimator so NOT ML

$$f(\mathbf{x};\beta) = \sum_{j=1}^{p} \beta_j x_j = \mathbf{x}^T \beta$$

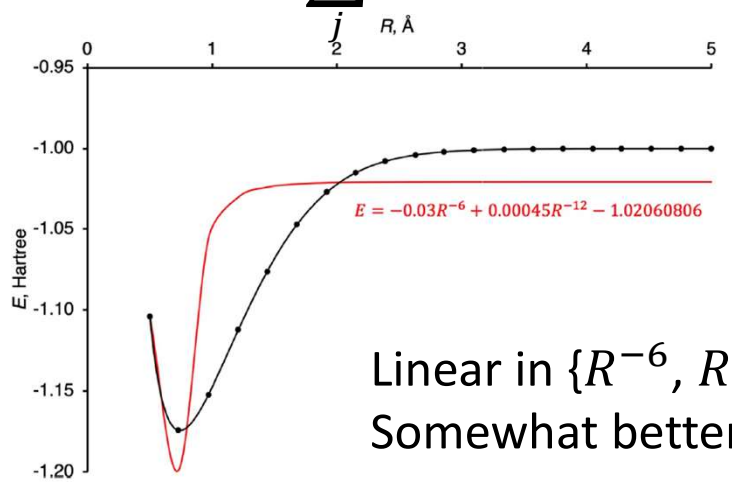
Example: Fit H_2 dissociation curve, $E(R)$

$$E(R;a,b) = aR + b$$



Linear in R
Bad fit

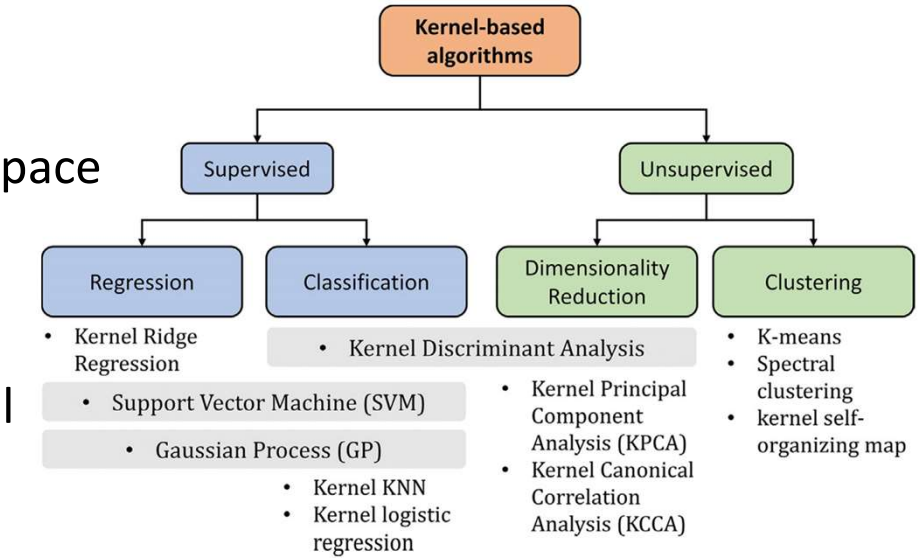
$$E(R;a,b,c) = \sum_{j}^{p} \beta_j \phi_j(R) = aR^{-6} + bR^{-1} + c$$



Linear in $\{R^{-6}, R^{-12}\}$ space
Somewhat better fit

Kernel methods

- based on similarity measure in high-dimensional space
- Extends linear fits to general non-linear models

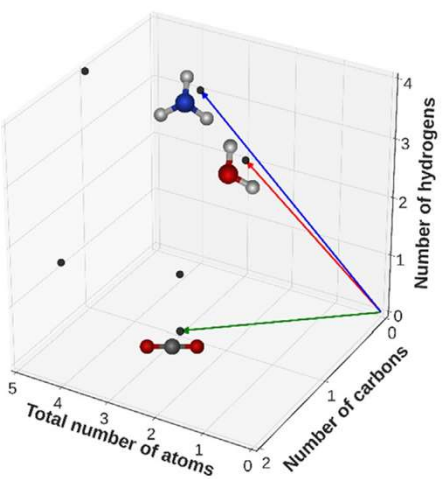


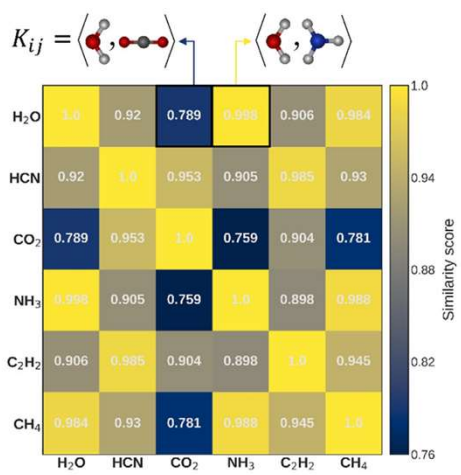
Molecule	num_atoms	num_carbons	num_hydrogens
H2O	3	0	2
HCN	3	1	1
CO2	3	1	0
NH3	4	0	3
C2H2	4	2	2
CH4	5	1	4

In this example, the kernel is a simple dot product (cosine similarity)

$$\boldsymbol{e}_{\mathrm{CO_2}} \cdot \boldsymbol{e}_{\mathrm{H_2O}} = \cos\theta$$

Kernels measure similarities ("distances")





Examples of types of Kernels

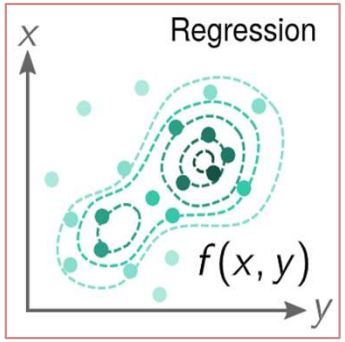
Linear kernel (dot product)

$$k(\mathbf{x}_i, \mathbf{x}') = \langle \mathbf{x}_i, \mathbf{x}' \rangle$$

Gaussian kernel

$$k(\mathbf{x}_i, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\sum_j^p (x_{ij} - x'_j)^2\right)$$

Kernel Ridge Regression



Step 1:
Start from linear regression

$$f(\mathbf{x};\beta) = \sum_{j=1}^{p} \beta_j x_j = \mathbf{x}^T \beta$$

Step 2:
Expand coeffs. in high-dimensional space spanned by training data

$$\beta_j = \sum_{i=1}^{N_{\mathrm{tr}}} \alpha_i x_{ij} \longrightarrow f(\mathbf{x}') = \sum_{i=1}^{N_{tr}} \alpha_i \langle \mathbf{x}_i, \mathbf{x}' \rangle$$

Dot product measures similarity between data point x_i and point of prediction x'

Step 3:
Do the same in space of nonlinear basis functions $\{\phi\}$

$$f(\phi(\mathbf{x}')) = \sum_{i=1}^{N_{tr}} \alpha_i k(\mathbf{x}_i, \mathbf{x}')$$

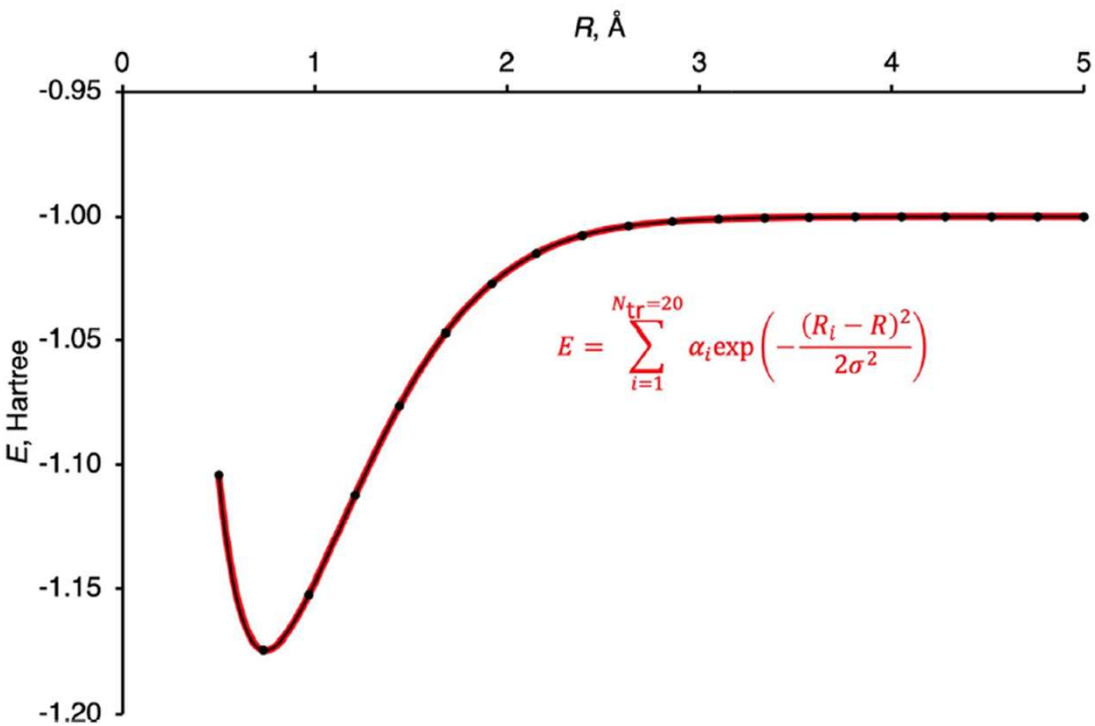
This is called the "**kernel trick**"
Many different kernels/basis functions:
Gaussian, Laplacian, polynomial, …

If $k(\mathbf{x}_i, \mathbf{x}') = \langle \mathbf{x}_i, \mathbf{x}' \rangle$

then we are back to linear regression but in training data space that can be expanded
-> Universal approximator

Example: Fit H_2 dissociation curve, $E(R)$ with Kernel Ridge Regression (KRR)

Gaussian kernel $$k(\mathbf{x}_i, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2}\sum_j^p \left(x_{ij} - x'_j\right)^2\right)$$



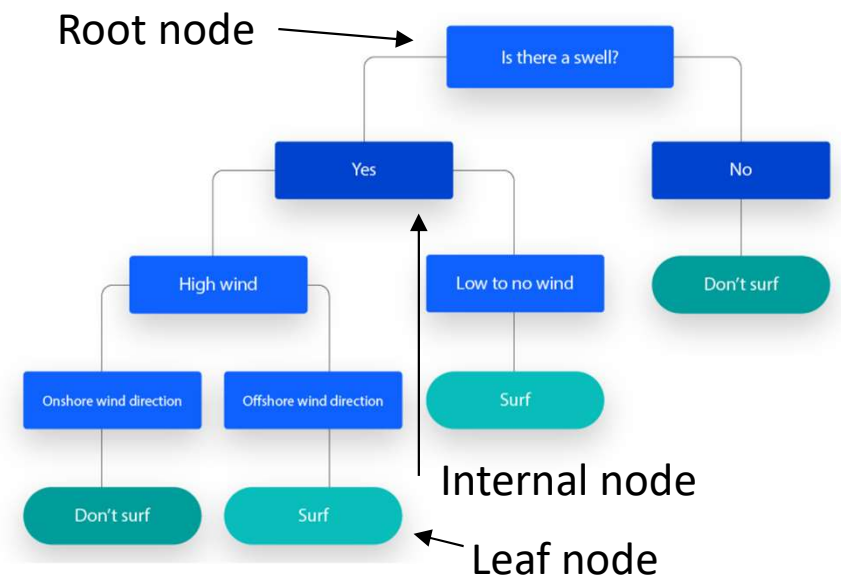
"Gaussian KRR puts Gaussian basis function with width σ on each data point and multiplies it with coefficient α_i."

We fit coefficients α_i
(least squares fit)

This is now a universal ML method. There is a risk of overfitting so we require regularization during fitting (fitting along the "Ridge" of solutions where coefficients remain as small as possible)

Decision Trees

- ✓ Non-parametric supervised learning methods for classification and regression
- ✓ Simple to understand and to visualize
- ✓ Can handle numerical and categorical data
- ○ Predictions are non-smooth
- ○ Large trees can be unstable -> overfitting
- ➢ Tackle via ensembles of trees -> Random Forests



Start with dataset of (**X**, Y) where Y is the label (surf,don't surf) and **X** is a set of attributes

swell (Y/N),
wind (numerical)
Wind direction (onshore/offshore)

Internal nodes represent attribute tests,
Branches represent attribute values
Leaf nodes represent final decisions or predictions

First find good attributes X -> Finding good feature representations

How are trees built? -> Information Gain or Gini Index

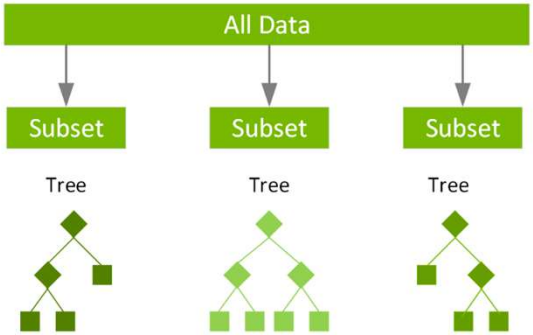
Decision Trees - Classification

How are trees built? -> identify attribute tests that maximize Information Gain or minimize Gini Index

Gini Index: measures how often a randomly chosen data point would be incorrectly identified by a certain attribute test.
Attributes with lower Gini indices are preferred as they better split the data

$$\text{Gini} = 1 - \sum_{i=1}^{n} p_i^2$$

p_i^2 probability of a certain outcome *i*



Ensembles of trees provide improved generalizability and robustness

- Gradient-boosted trees (e.g. XGBoost)
- Random forests

Reinforcement Learning

- **Agent** performs certain **actions** in an **environment** at each time step in a sequential decision-making framework
- Actions change the **state** and can provide positive or negative feedback -> goal is to learn a **policy** that provides maximum **reward**

- Uses rewards instead of labels to learn

- Temporal Credit Assignment Problem: associating a reward with an action
- Finding Trade-off between Exploitation vs. Exploration

Examples

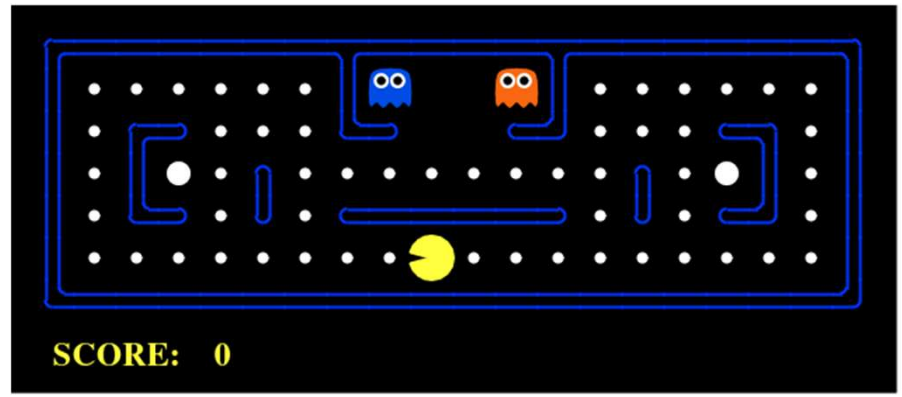
- Video games
- Training robots
- AlphaGo

Markov process

Sequence of states $s_1, s_2, s_3, \dots, s_t, \dots$

Transition probability $t \to t+1$

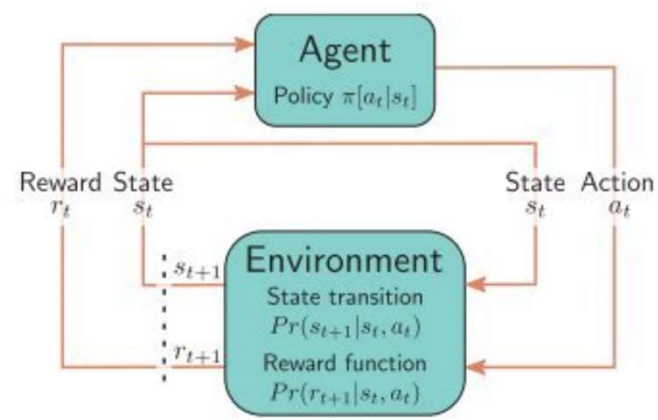
$$Pr(s_{t+1}|s_t)$$



Markov reward process

Sequence of rewards $r_1, r_2, r_3, \dots, r_t, \dots$ discount factor $\gamma^k \in (0,1]$

return $$G_t = \sum_{k=0}^{\infty} \gamma^k \, r_{t+l+1}$$



Agent performs actions a_t

Markov decision process

Sequence of states and actions $s_1, a_1, s_2, a_2, s_3, a_3, \dots, s_t, a_t, \dots$

Actions affect transition probability $Pr(s_{t+1}|s_t, a_t)$

Actions affect reward probability $Pr(r_{t+1}|s_t, a_t)$

Policy: $\pi[a|s]$ determines the action, stochastic or deterministic, stationary or time-dependent

Assign value functions to states, and actions to determine optimal reward G_t

State value function	$v[s_t \| \pi]$	Expected return for being in state t
Action value function	$q[s_t, a_t \| \pi]$	Expected return for executing action in state t

If we know the optimal action values, we can derive the optimal policy $\pi[a_t|s_t]$

$$v[s_t] = \sum_{a_t} \pi[a_t|s_t] q[s_t, a_t]$$

$$q[s_t, a_t] = r[s_t, a_t] + \gamma \cdot \sum_{s_{t+1}} Pr(s_{t+1}|s_t, a_t) v[s_{t+1}]$$

$\longrightarrow$ Bellman equations to define optimal policy

Types of RL

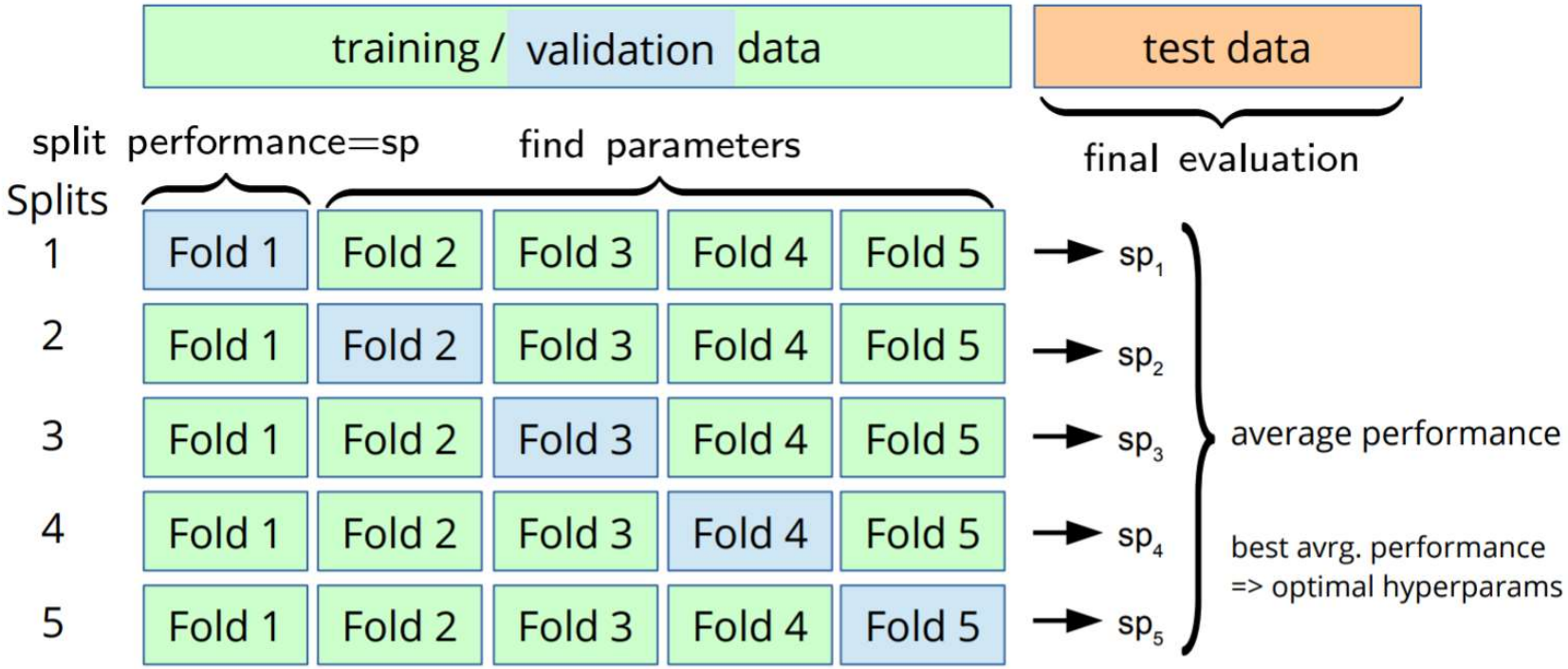
- Tabular reinforcement learning (methods that don't rely on function approximation/ML)
- Fitted Q-learning (action value function replaced by machine learning model)
- Policy gradient methods (directly learn a stochastic policy $\pi[a_t|s_t]$)
- and many more…

Putting it all together

Typical workflow in ML project

1. Define the task and the objective (informing the loss func. and data gen.)
2. Generate and clean the data (e.g. find patterns, find outliers) **Clustering**
3. Discover and visualize the data to gain insights (find correlations) **Classification**
4. Prepare data for training (e.g. train/test split, scaling) **Dimensionality Reduction**
5. Find good data representation: "Featurization"
6. Select, train, and evaluate model (e.g. calculate MAE, RMSE) **e.g. Classification or Regression**
7. Optimise and fine-tune model with cross-validation
8. Assess <u>accuracy</u> and <u>uncertainty</u> of trained model
9. Generate more data to improve accuracy/uncertainty (e.g. active learning)
10. When ready, deploy model (£££ and/or manuscript) **Clustering**

Validating your model: K-fold Cross Validation



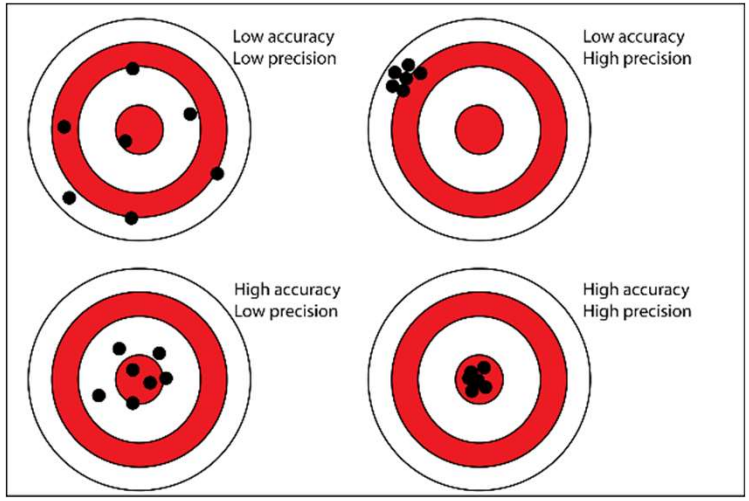
K-fold CV

- used for model validation (calculate accuracy and standard deviation of prediction)
- used for Hyperparameter optimisation (Grid search, Random search)
- Avoids overfitting
- Can help to identify outliers and unbalanced datasets

Uncertainty Quantification

Accuracy of prediction: MAE or RMSE

Precision of prediction: ?????

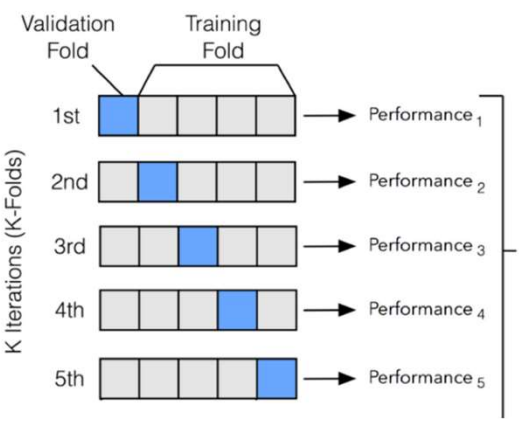


2 Sources of uncertainty in prediction:

- Aleatoric uncertainty (statistical error, noise in data)
- Epistemic uncertainty (intrinsic to model)

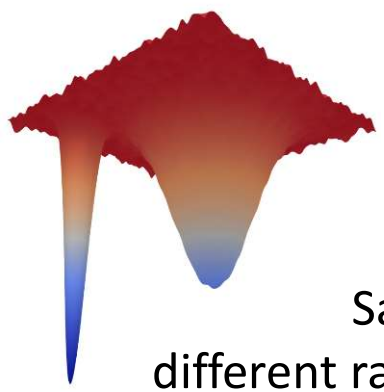
How to calculate uncertainty / standard deviation?

Bootstrapping (e.g. subsampling)



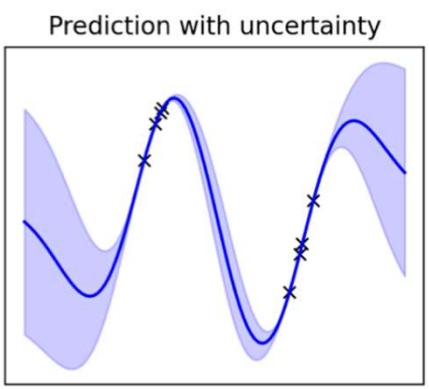
Mean & StDev

Ensemble Learning (stacking, bagging, boosting)



e.g. **Random Ensembles**
Same data,
Same model,
different random seed

Bayesian Uncertainty (e.g. Gaussian Process Regression)



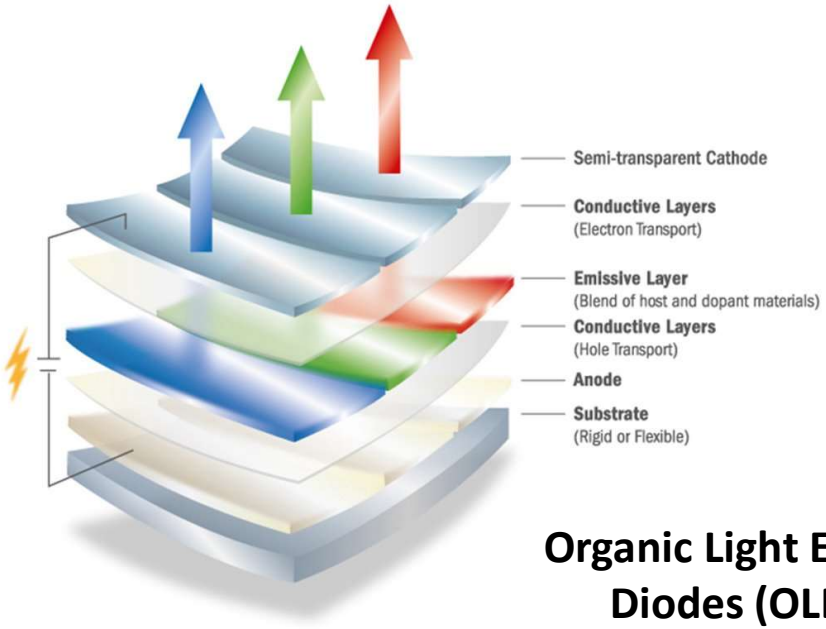
Challenges in Machine Learning

Challenge	Techniques to address
• Insufficient quantity of training data →	Adaptive/Active Learning
• Nonrepresentative training data (Bias) →	Uncertainty Quantification
• Poor-Quality data (Noise) →	Feature Engineering/Selection
• Irrelevant Features →	Hyperparameter Optimisation
• Overfitting →	

Research Example: Generative molecular design

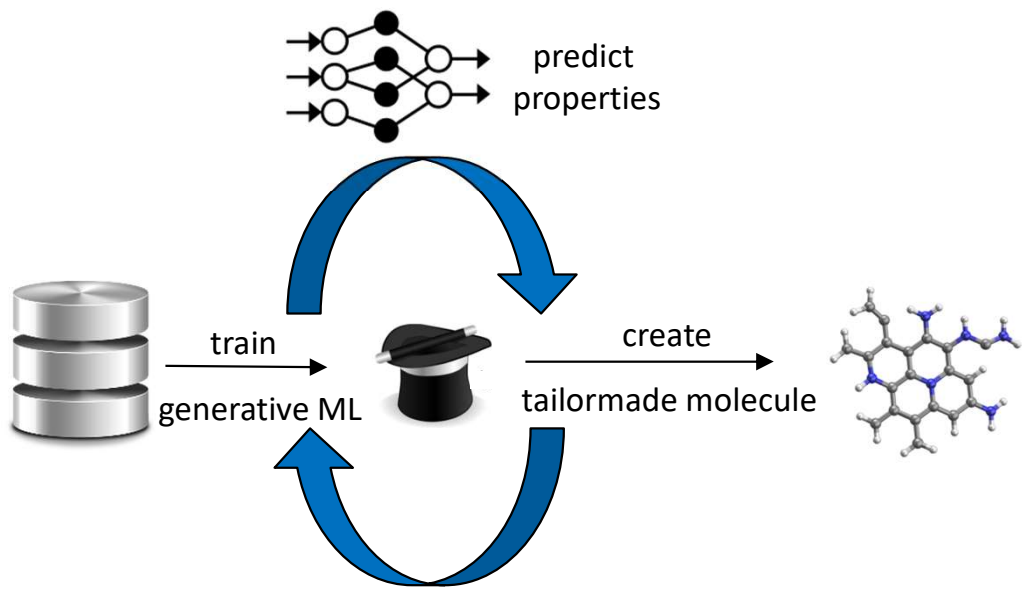
Westermayr, Maurer, Chem. Sci. 12, 10755 (2021)
Westermayr et al., Nature Computational Science 3, 139-148 (2023)
Koczor-Benda et al, arXiv: 2503.14748
Koczor-Benda et al., arXiv: 2503.21328

Example: Designing molecules with tailormade properties?

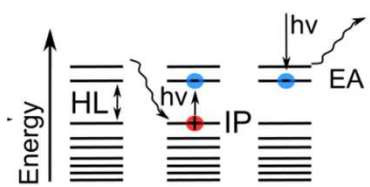


Organic Light Emitting Diodes (OLEDs)

"Artificial molecular forge"



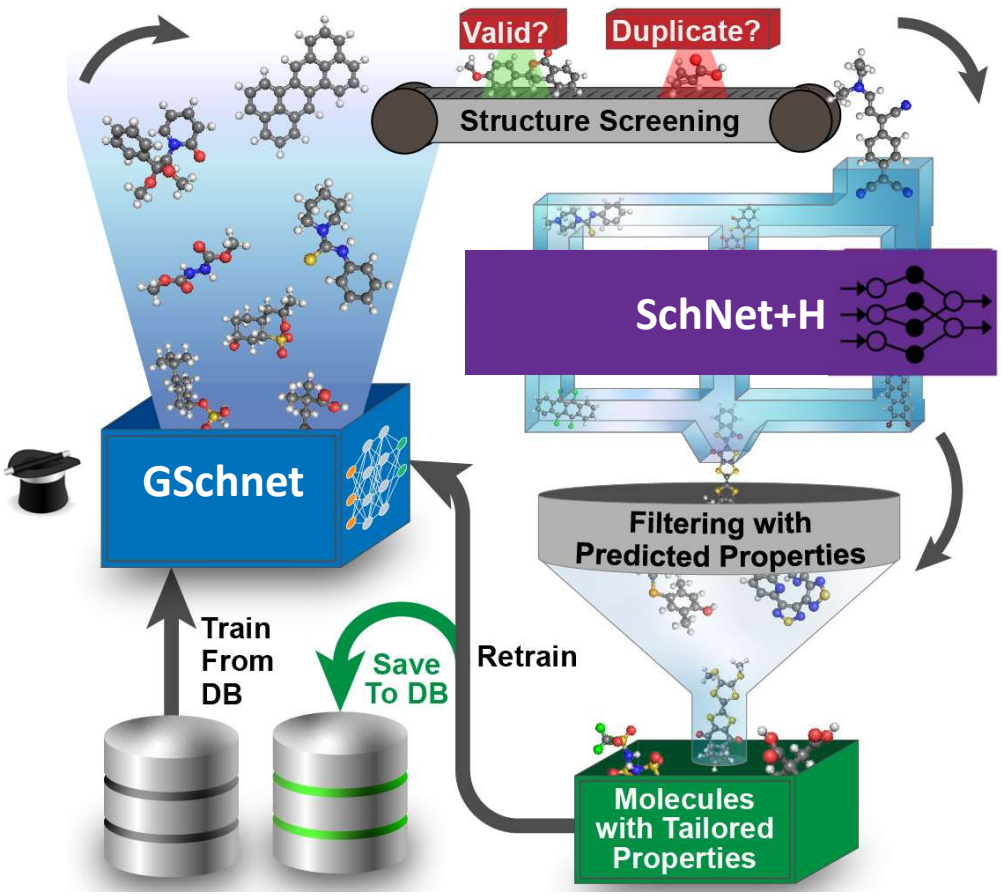
Application:
Generative molecular design of organic electronics



IP ... Ionization potential
EA ... Electron affinity
HL ... HOMO-LUMO gap

Nature Computational Science 3, 139–148 (2023)

Generative Design of Molecules with Tailored Properties

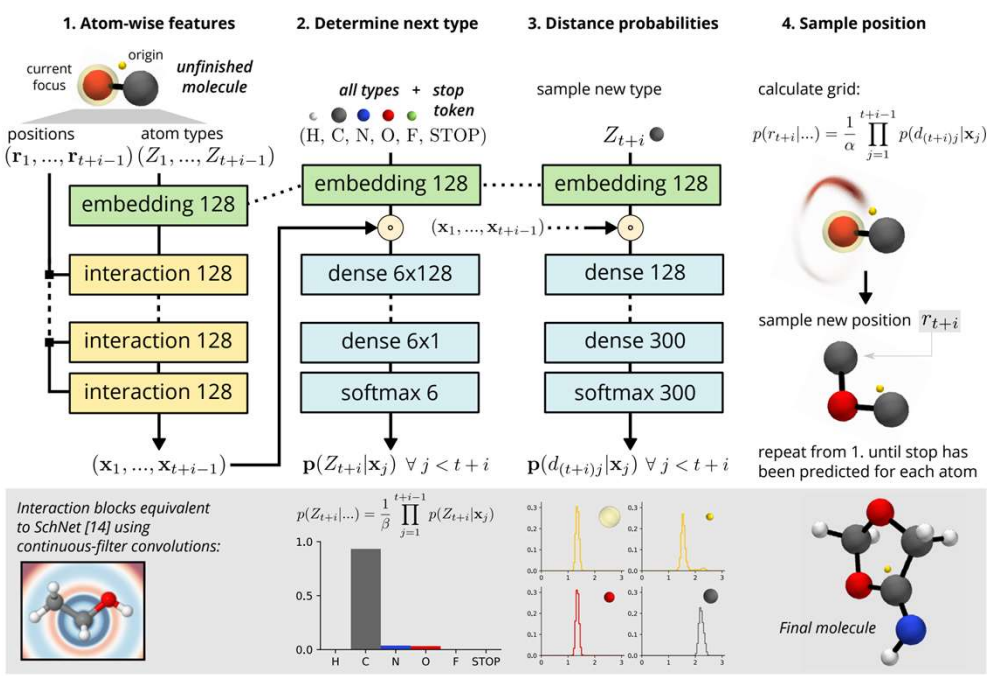


Nature Computational Science 3, 139–148 (2023)

Generative deep learning of 3D molecular structures

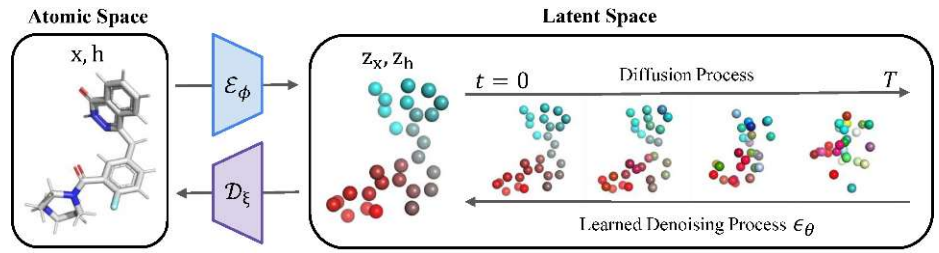
Autoregressive atom-by-atom construction of molecules

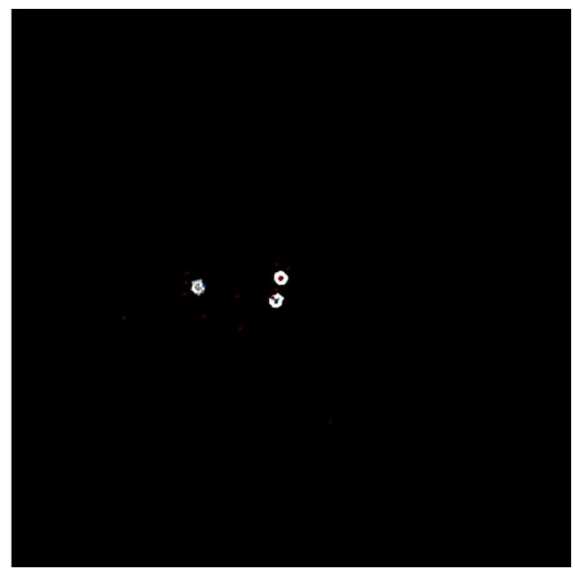
GSchnet



G-SchNet model: N. Gebauer et al. NeurIPS 32 (2019)

Geometric diffusion models





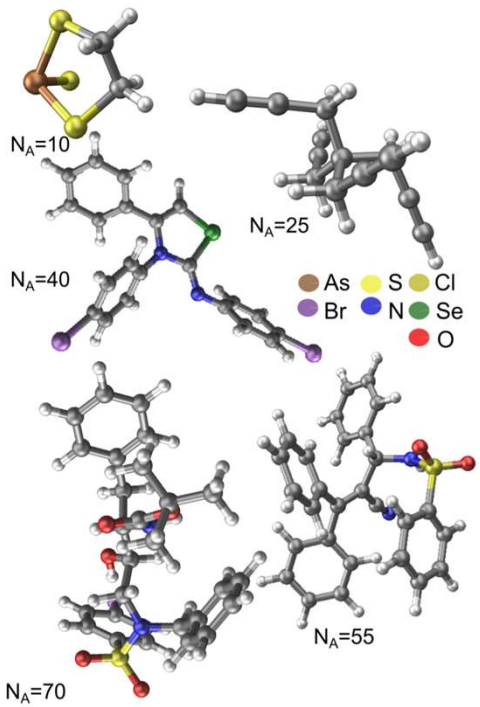
Model trained by MChem student Abudalla Al-Fekaiki

GeoLDM model: Xu et al. ICML (2023)

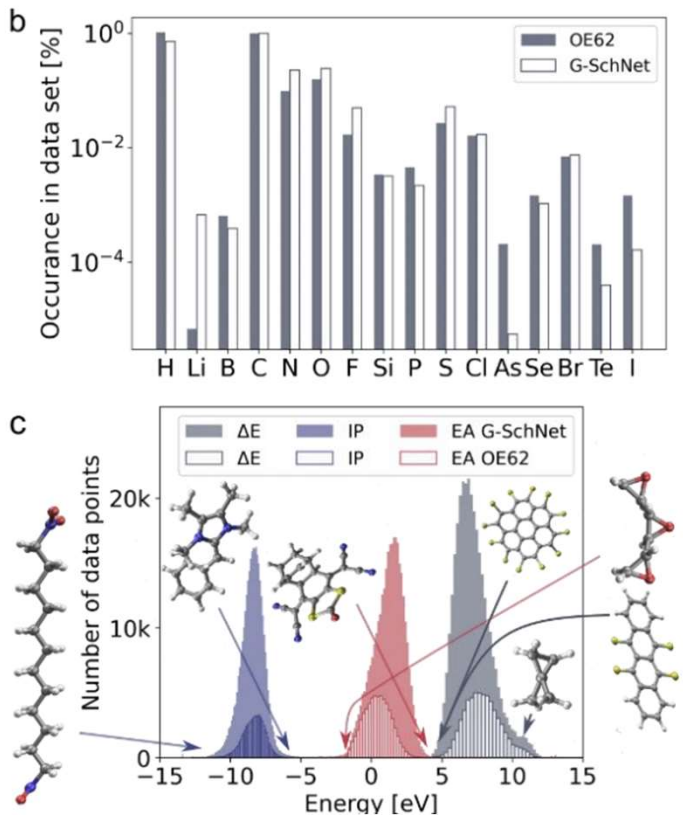
Generative deep learning of 3D molecular structures

GSchnet

trained on
OE62 database



Generated structures

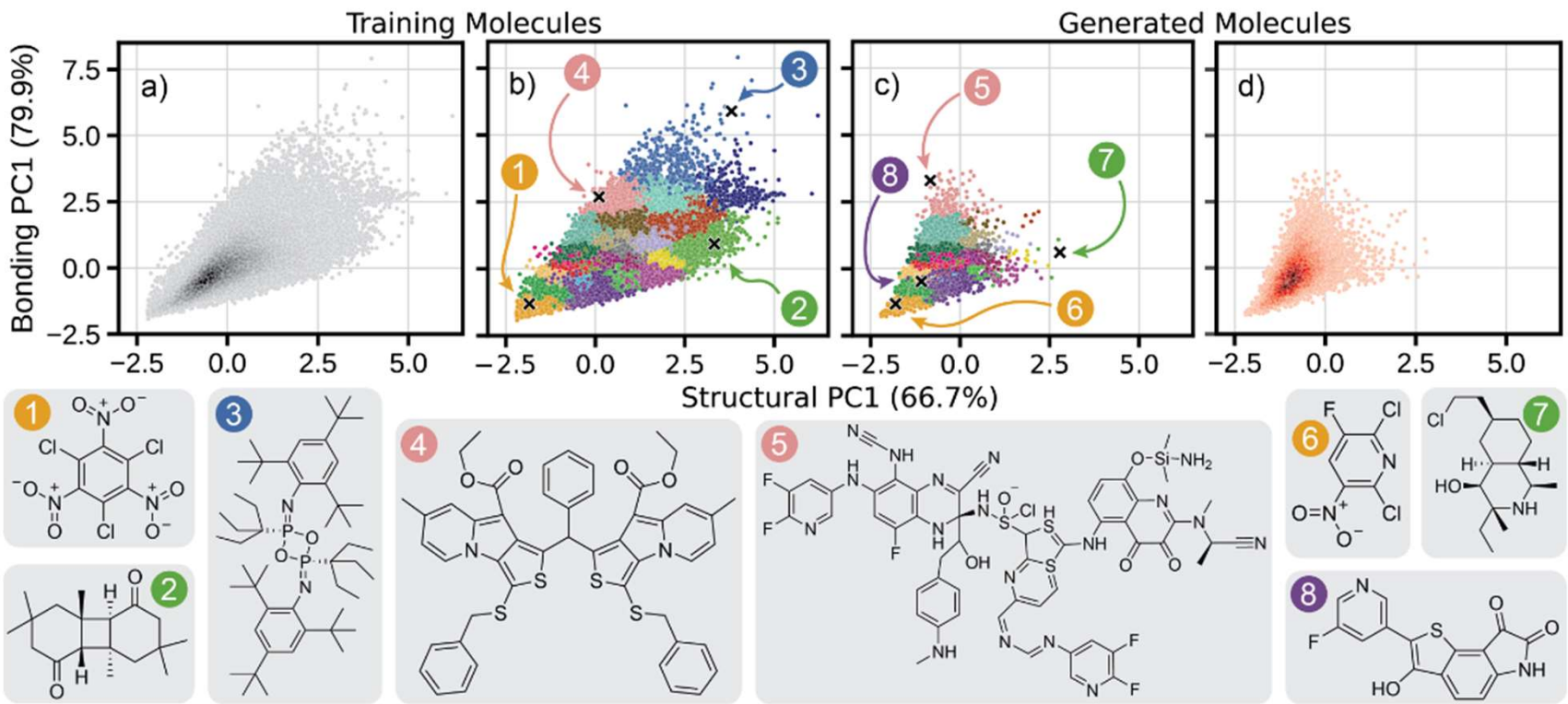


reproduce
elemental
distribution

reproduce
property
distribution

Nature Computational Science 3, 139–148 (2023)

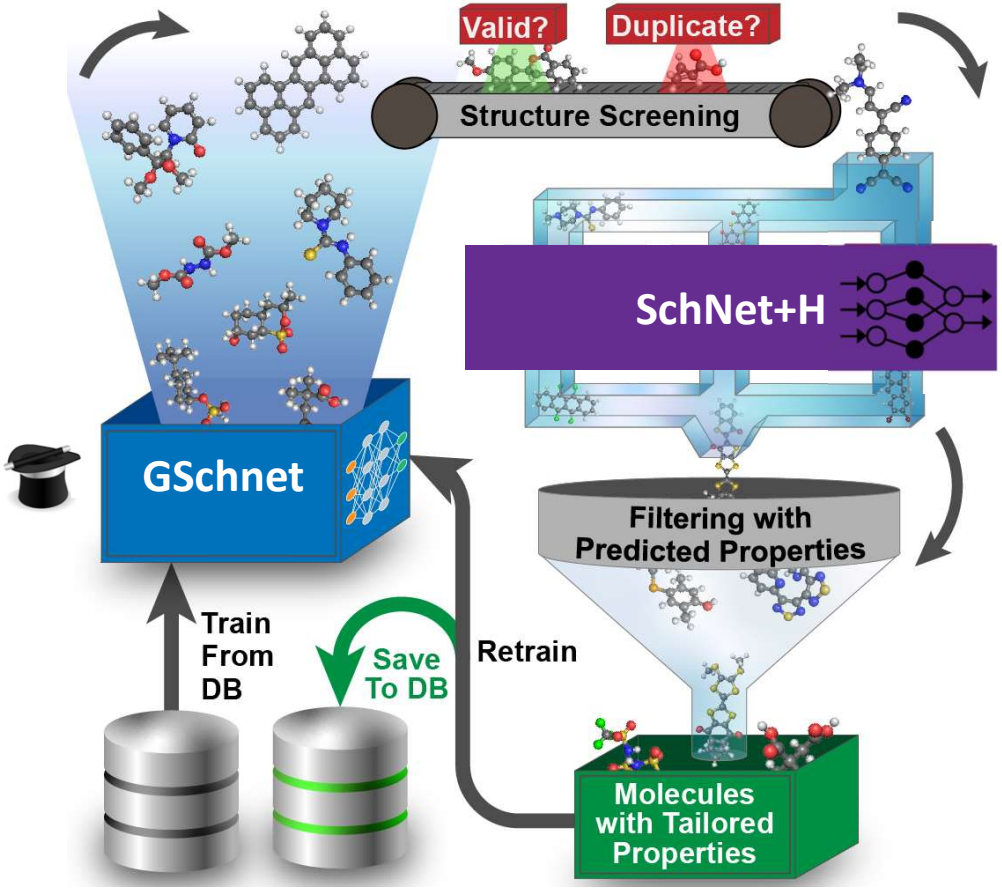
Bias in generated molecules!

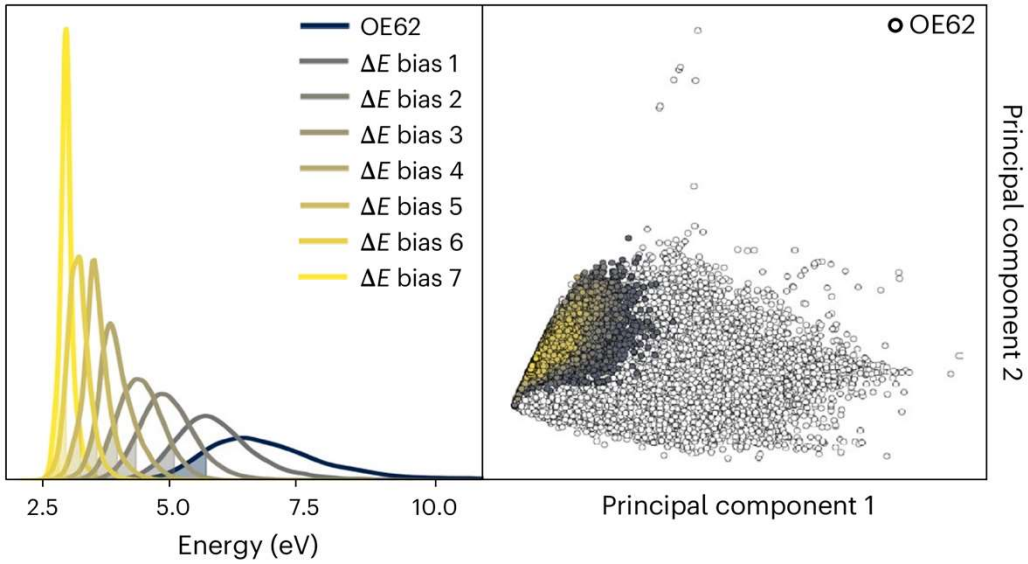


Model misses out on saturated/aliphatic structures

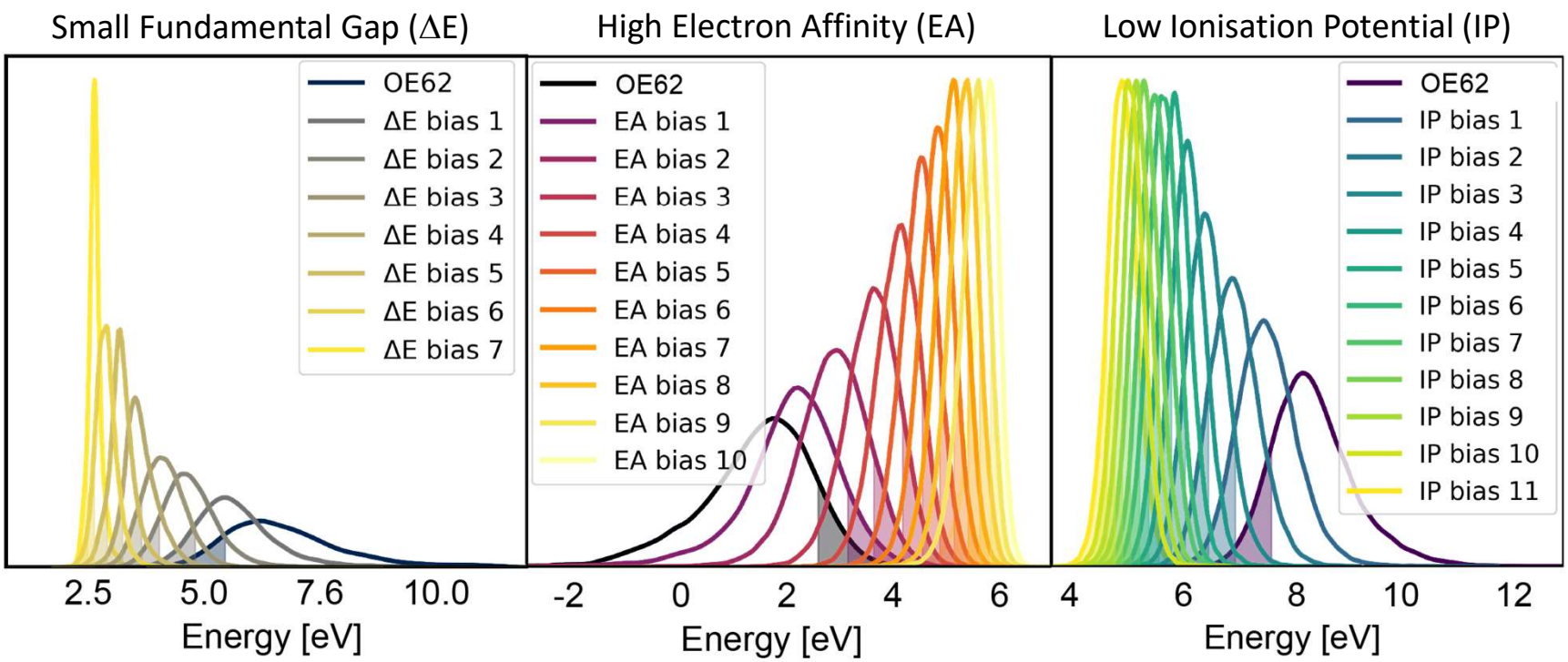
Koczor-Benda et al., arXiv: 2503.21328

Generative Design of Molecules with Tailored Properties



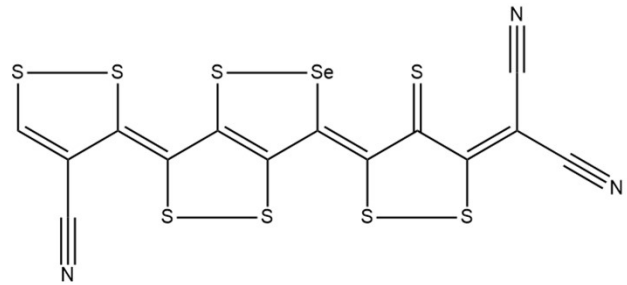


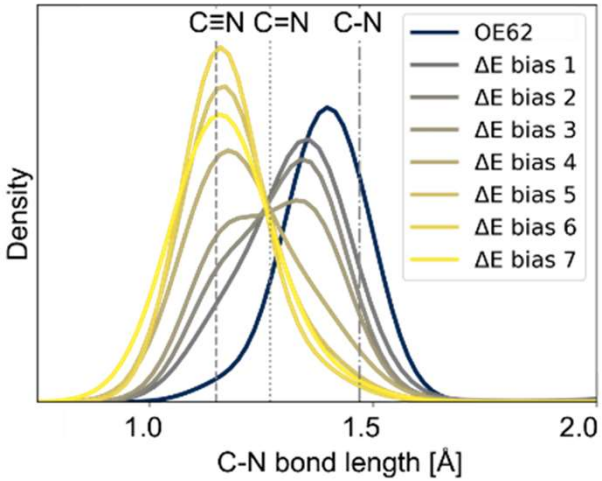
Nature Computational Science 3, 139–148 (2023)

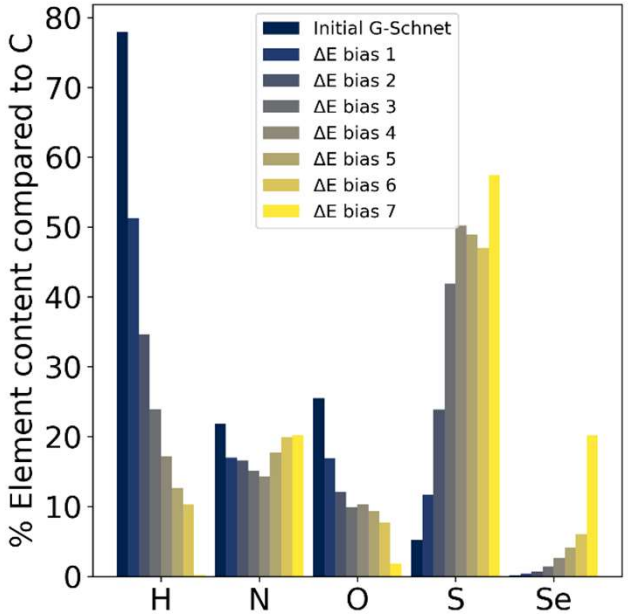

Small Fundamental Gap (ΔE)
High Electron Affinity (EA)
Low Ionisation Potential (IP)
OE62
ΔE bias 1
ΔE bias 2
ΔE bias 3
ΔE bias 4
ΔE bias 5
ΔE bias 6
ΔE bias 7
EA bias 1
EA bias 2
EA bias 3
EA bias 4
EA bias 5
EA bias 6
EA bias 7
EA bias 8
EA bias 9
EA bias 10
IP bias 1
IP bias 2
IP bias 3
IP bias 4
IP bias 5
IP bias 6
IP bias 7
IP bias 8
IP bias 9
IP bias 10
IP bias 11
2.5
5.0
7.6
10.0
-2
0
2
4
6
4
6
8
10
12
Energy [eV]
Energy [eV]
Energy [eV]

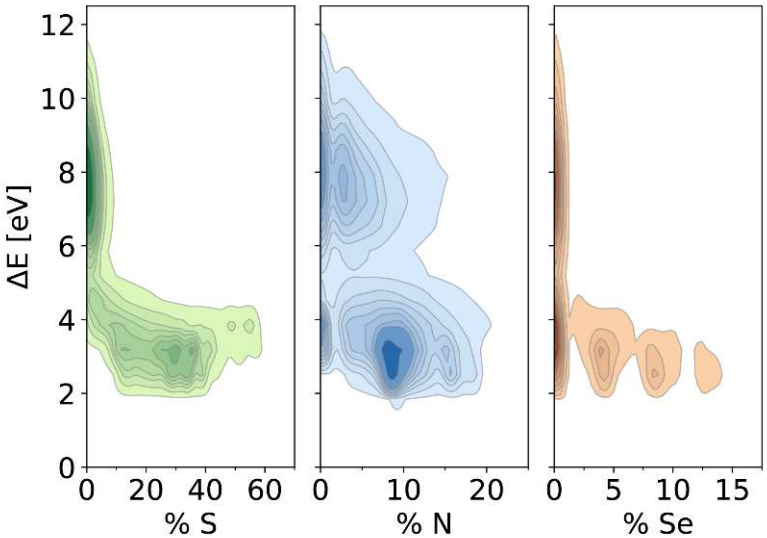
Bonding Descriptor Trends

Small Fundamental Gap (ΔE) Biasing

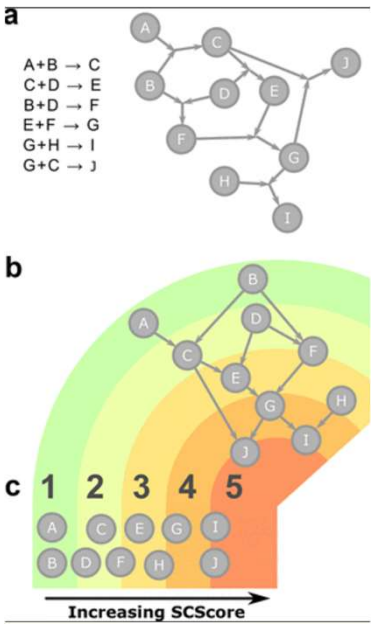




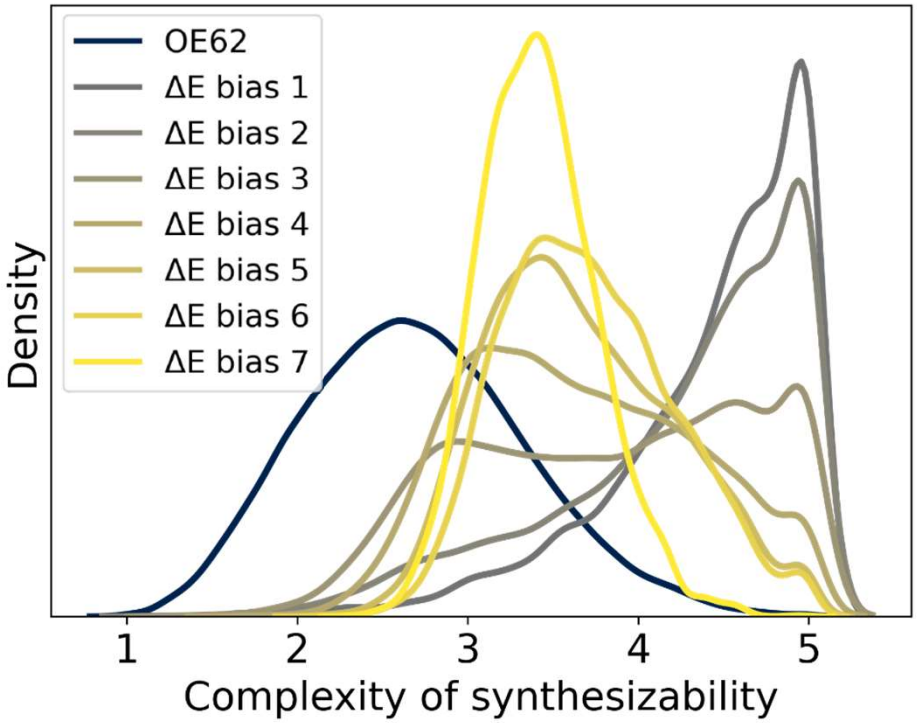




Synthetic Viability

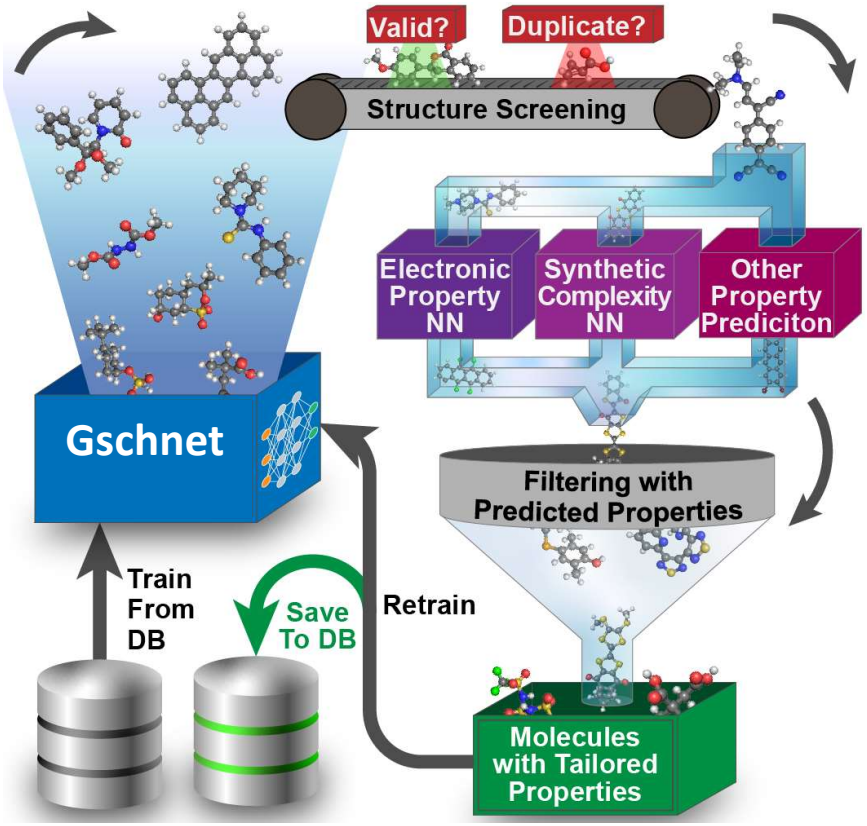


- High selenium content leads to molecules that are difficult to synthesise.
- We quantify this with the SCScore metric.[1]

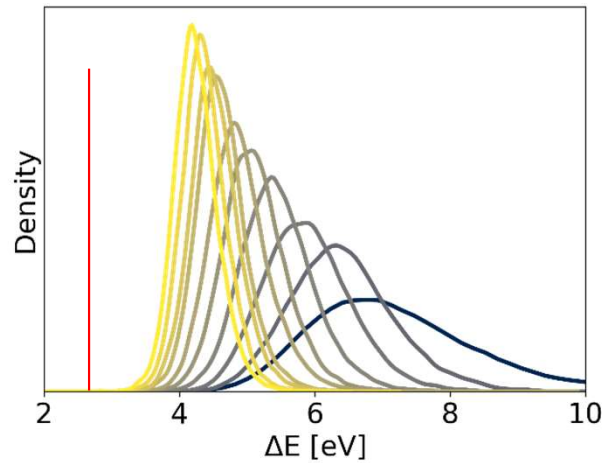


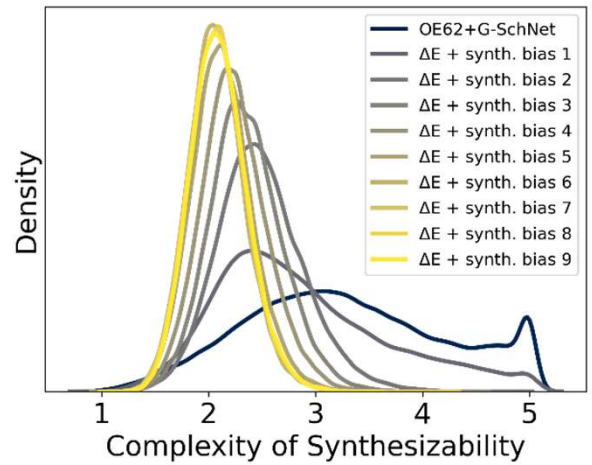
SCScore: Coley et al, J. Chem. Inform. Model. 58, 252-261 (2018)

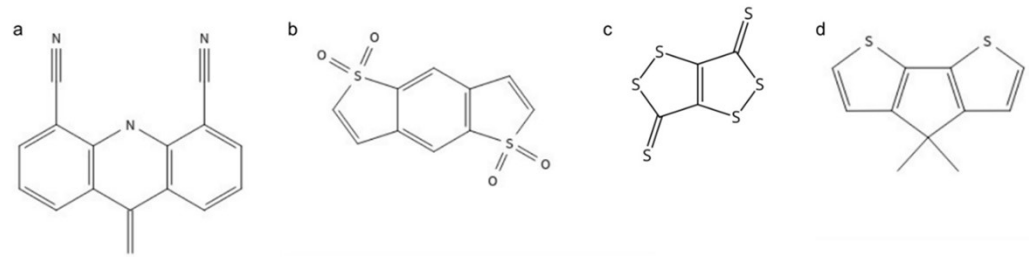
Generative Design of Molecules with Multi-Property Optimisation



Combine electronic screening and synthetic complexity screening



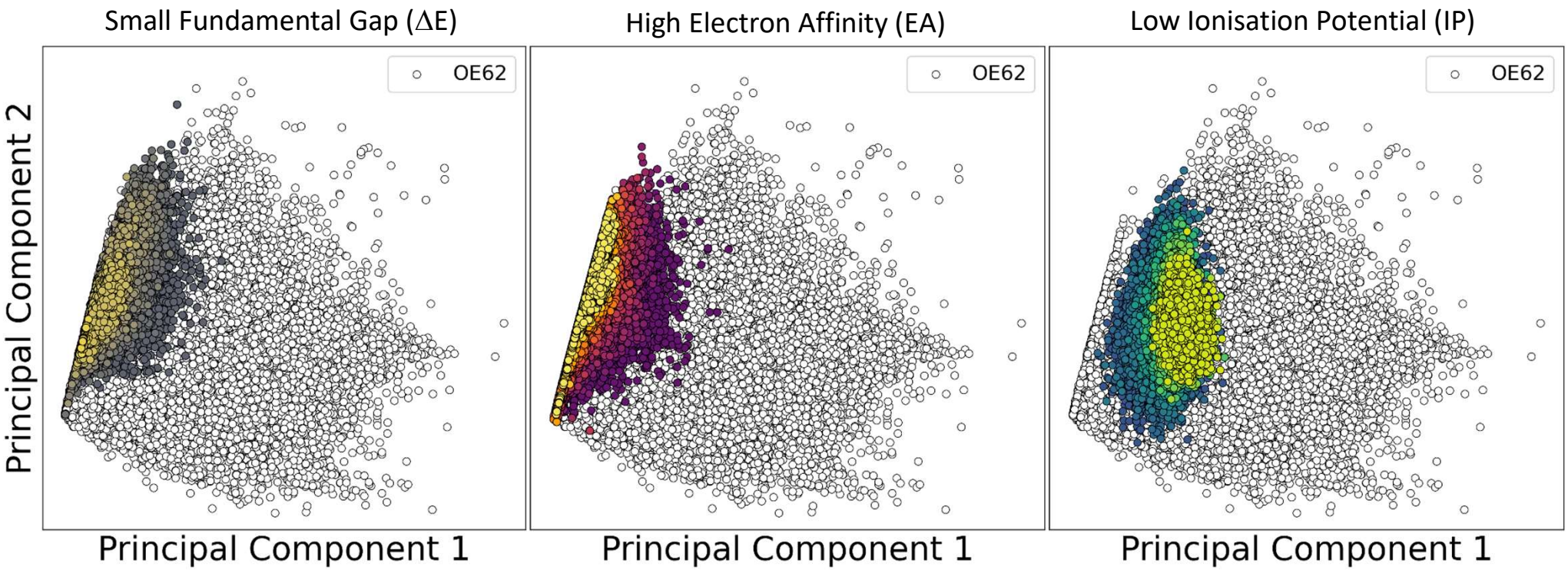




Nature Computational Science 3, 139–148 (2023)

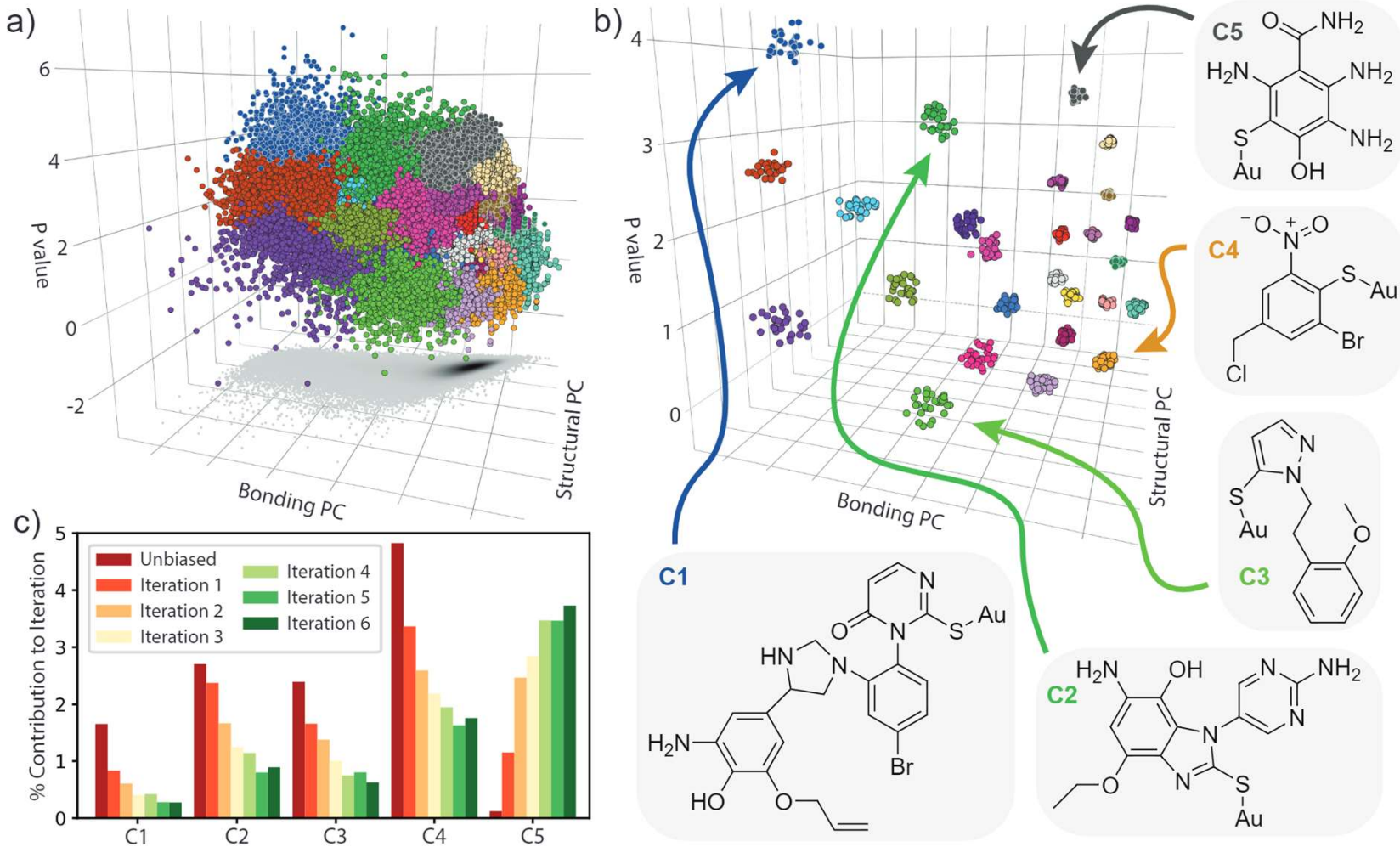
Interpreting the Data

Latent "Chemical Space" Maps with Principal Component Analysis



https://github.com/maurergroup/gschnettools

Interpreting the Data: Clustering



Koczor-Benda et al, arXiv: 2503.14748

Thank you!

Go and use ML methods for your research!
BUT PLEASE

- Remember: learning ≠ understanding
- Embrace reproducibility (clear workflows, write tutorials)
- Embrace openness (publish your models, data & scripts!)