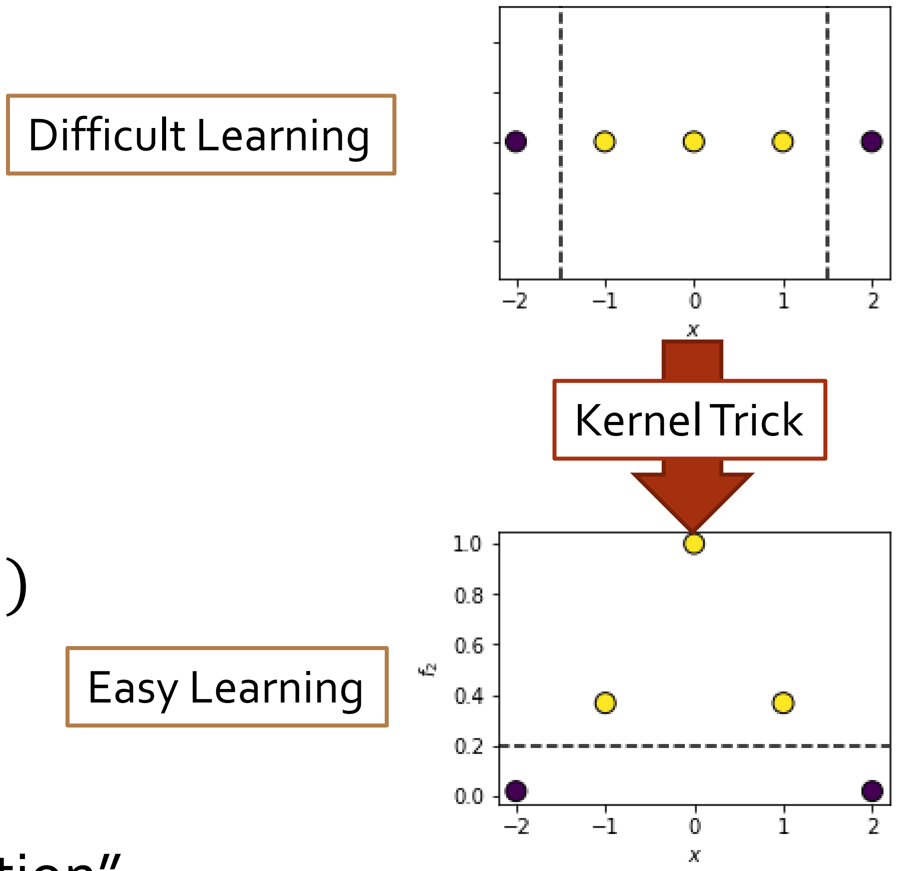
KERNEL METHODS FOR MOLECULAR MACHINE LEARNING

Logan Ward
Asst. Computational Scientist
Argonne National Laboratory

24 January 2022

What are kernel methods?

The “kernel trick” is to change data from
low to high dimension space
using a **pairwise similarity** “kernel” function.



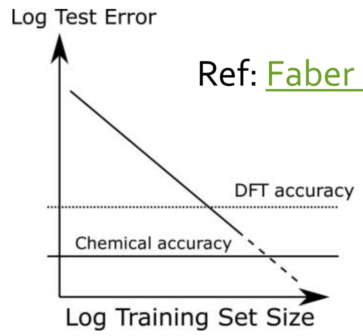
Key terms:

- Kernel function: $k(x_i, x_j)$
- Kernel matrix: $K_{ij} = k\left(x_i, x_j\right)$
- Kernel Ridge Regression: $f(x_i) = K\alpha = \sum_j \alpha_j k(x_i, x_j)$

Main Concept: “Make learning simple with a kernel function”

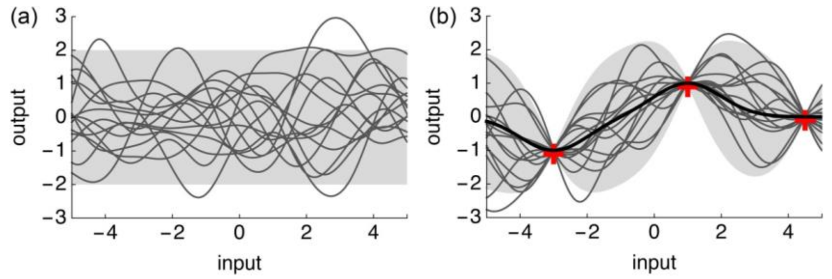
Kernel Methods have great properties

- Instance based learning: Complexity grows with data

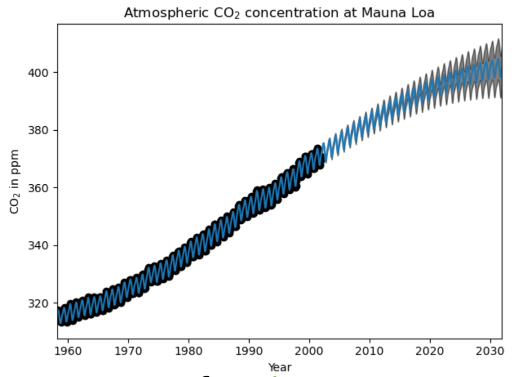


Ref: Faber et al. JCTC. (2017)

- Good uncertainty methods (GPR)



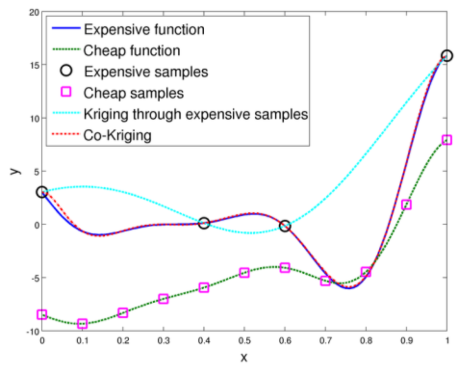
Ref: Rupp. Int. J. Quant. Chem., (2015)



Ref: Scikit-Learn Docs

- Flexibility in kernels: different kinds of non-linearity

- Multi-resolution methods



Ref: Zhang *et al.* AIAA 2013. (2013)

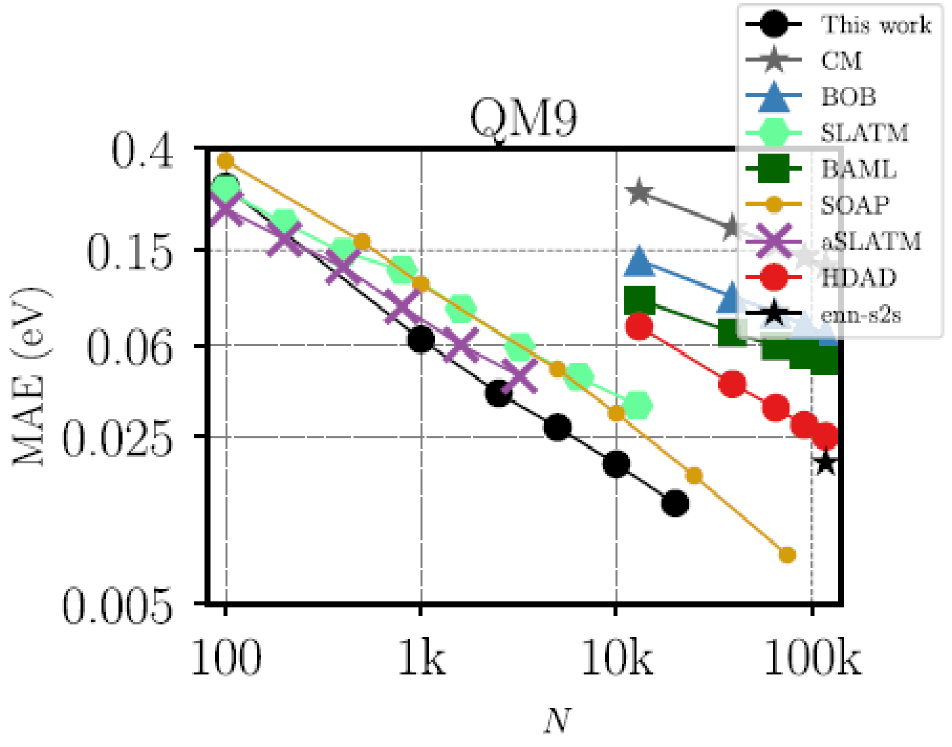
How do I make similarity functions for molecules?

This question has been studied for 10+ years

What kind of physics do we put into the kernel

- Interpolation between changes of atom type
- Sensitivity to small changes in environments
- Invariance to rotation, translation, permutation
- …

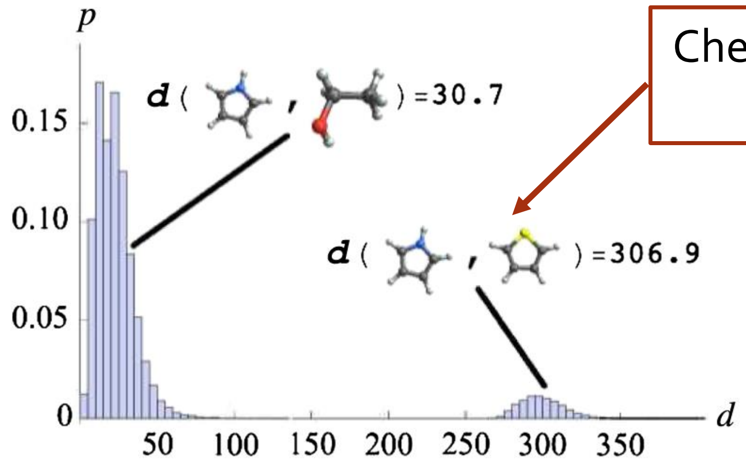
There are a half dozen (or more*) ways to do this



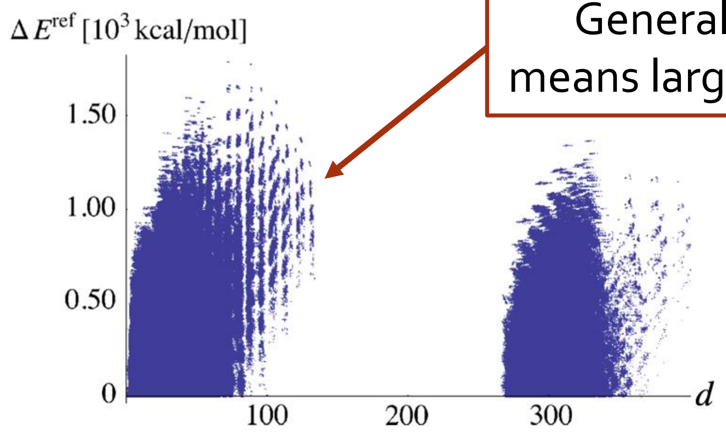
Ref: Faber et al. JCP. (2018)

*Follow Anatole von Lilienfeld's group

Case Study: Coulomb Matrix



Chemical changes create large distances



Generally, larger distance means larger different in energy

Simple formula:

$$M_{ij} = \begin{cases} 0.5 Z_i^{2.4} & i = j \\ \frac{Z_i Z_j}{\left\| R_i - R_j \right\|_2} & i \neq j \end{cases}$$

Captures atomic positions and types

Problem: M not permutation invariant

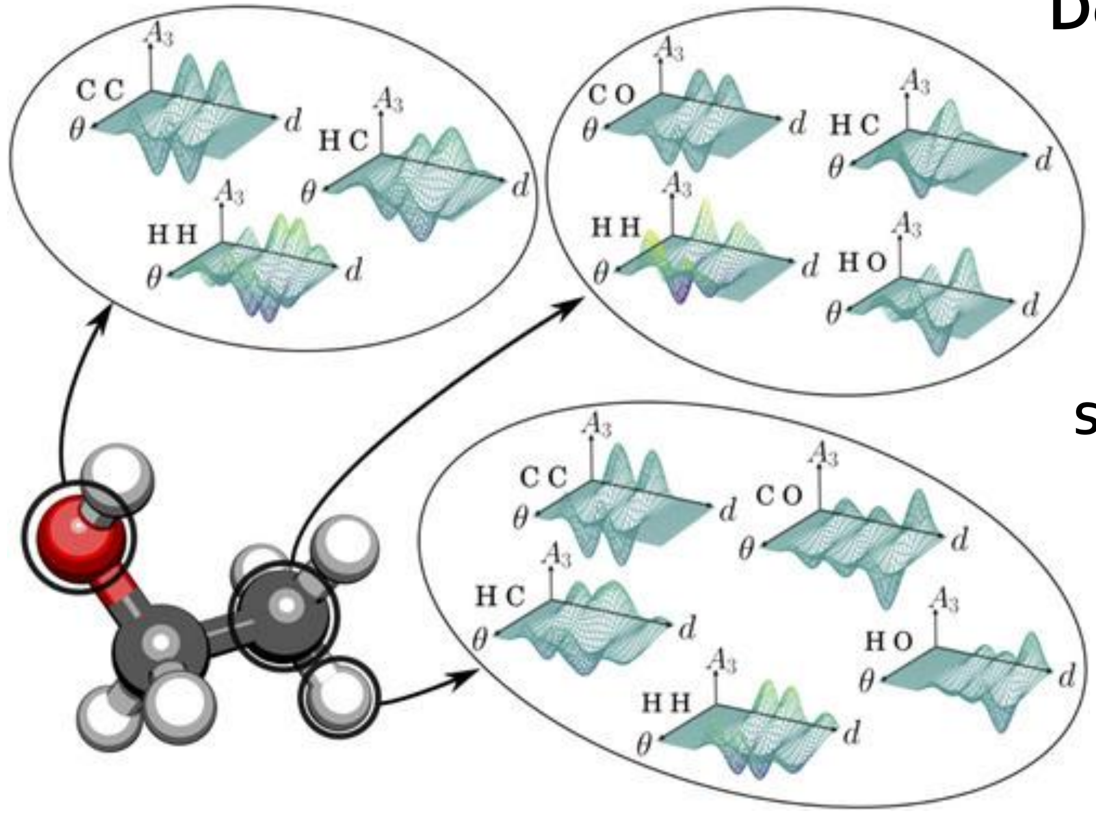
Solution: Use eigenvalues of M (ϵ)

$$d_{ij} = \left\| \epsilon_i - \epsilon_j \right\|_2$$

$$k(x_i, x_j) = e^{\frac{-d_{ij}}{\sigma}}$$

Ref: Rupp et al. PRL (2012)

Case Study: FCHL



Describes atoms using…

- **"alchemical"** - difference based on period, group
- **"many body"** - Capture bond distances and angles
- **"distributions"** - As gaussian functions

Similarity between atoms are computed with overlap integrals

$$A_1(x, y; I) = \mathrm{e}^{-\frac{(\mathrm{P_I - x})^2}{2\sigma_P^2} - \frac{(G_I - y)^2}{2\sigma_G^2}}$$

$$\Delta\big(A_1(I), A_1(J)\big) = \iint \big(A_1(I) - A_1(J)\big)^2 dxdy$$

$$= \frac{1}{2}\exp\left(-\frac{\left(P_I - P_J\right)^2}{4\sigma_P^2} - \frac{\left(G_I - G_J\right)^2}{4\sigma_G^2}\right)$$

Ref: Faber et al. JCP (2018)

Scalable Kernels: What and Why

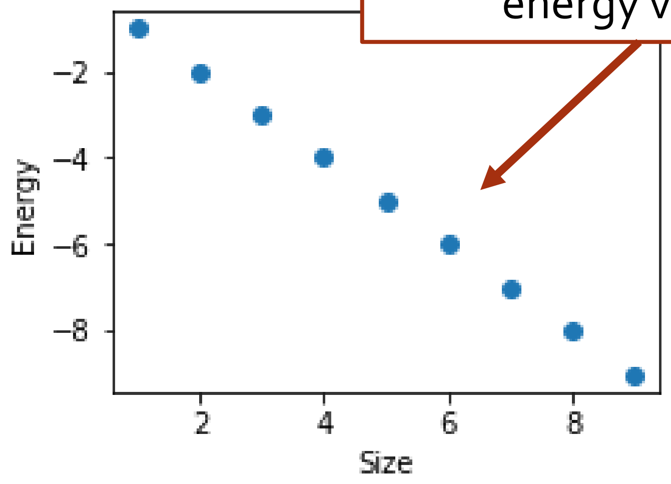
Problem: FCHL has *atomic similarity* but we want *molecular properties*

Solution: Make a “scalable” kernel that encodes atomic -> molecular relationship

Example: Energy is often a sum over atoms.

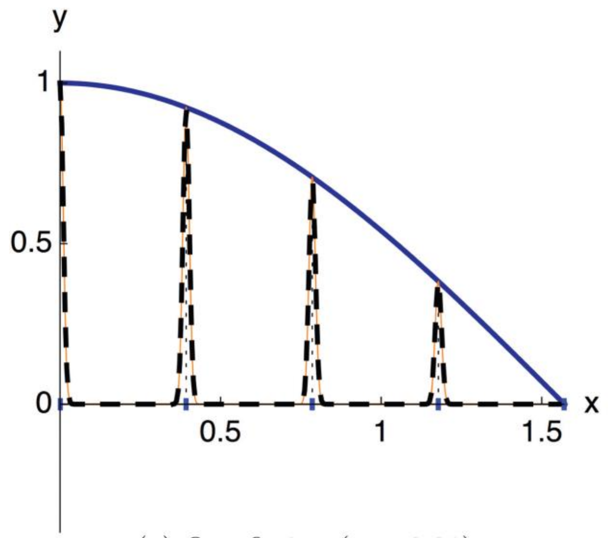
$$K_{mol}(x, y) = \sum_i \sum_j k_{atom}(x_i, y_j)$$

Automatically captures energy vs size

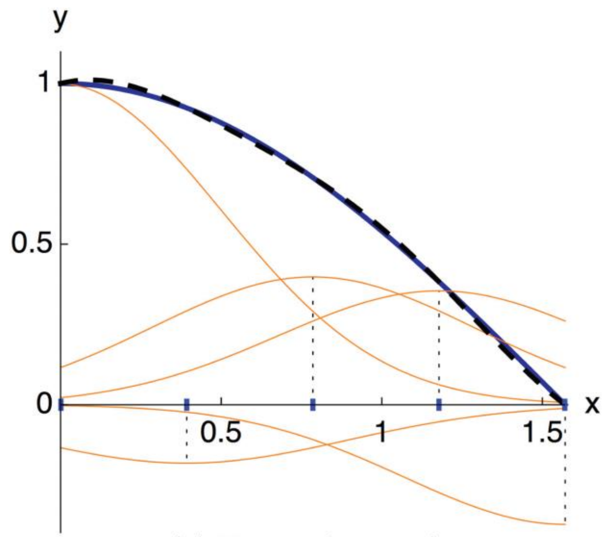


Full explanation: “FCHL in one notebook”

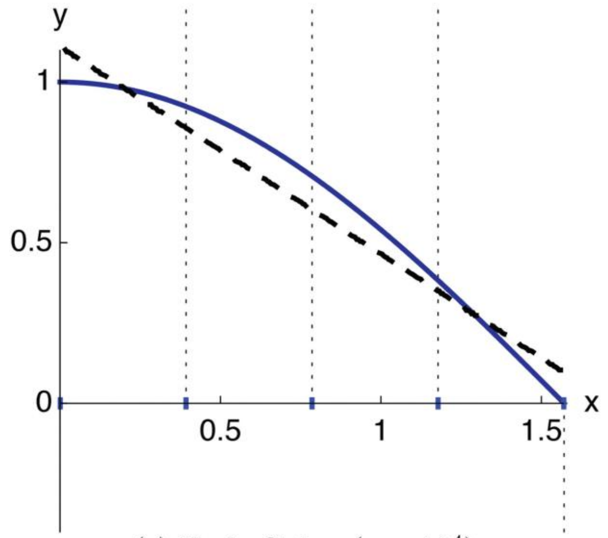
Key issue: Adjusting hyperparameter is *very* important



(a) Overfitting ($\sigma = 0.01$)



(b) Fitting ($\sigma = 0.5$)



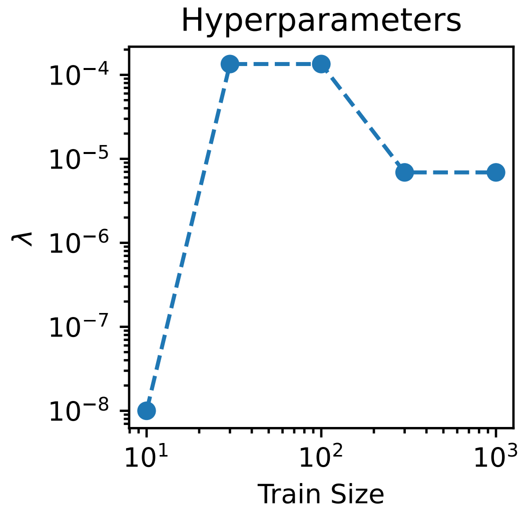
(c) Underfitting ($\sigma = 10^4$)

$$\boldsymbol{x} = \{0, \tfrac{1}{8}\pi, \tfrac{2}{8}\pi, \tfrac{3}{8}\pi, \tfrac{4}{8}\pi\}$$

	α_1	α_2	α_3	α_4	α_5
(a)	1.000	0.924	0.707	0.383	0.000
(b)	0.998	−0.182	0.398	0.355	−0.372
(c)	-10^{13}	10^{12}	10^{13}	10^{12}	-10^{12}

(d) Training points $\boldsymbol{x}$ and weights $\boldsymbol{\alpha}$

You cannot be sure if a kernel method works until after investing significant time

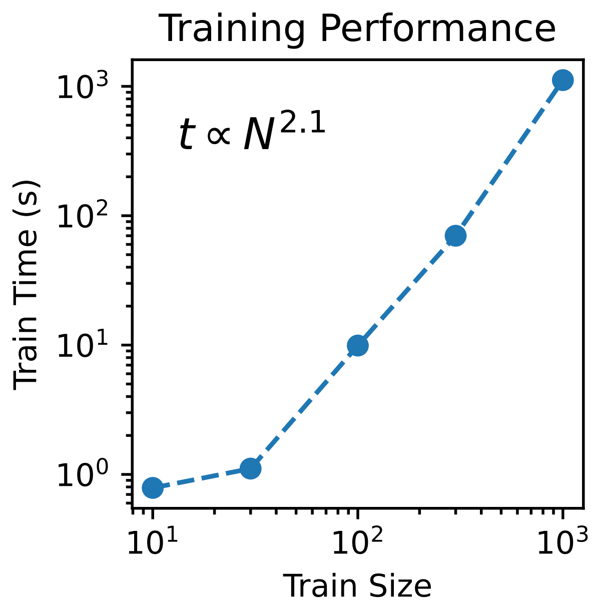


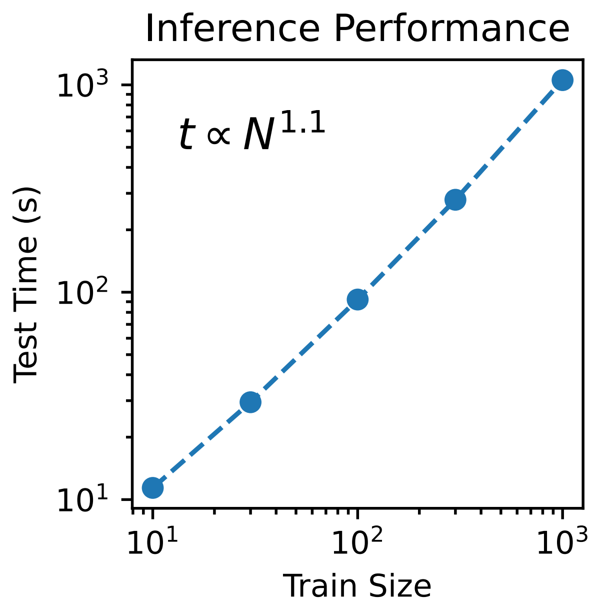
Effort must be replicated as training set changes

Ref: Rupp. Int J. Quat. Chem. (2015)

Dark side of kernel methods: Scaling

Fitting KRR models is expensive…





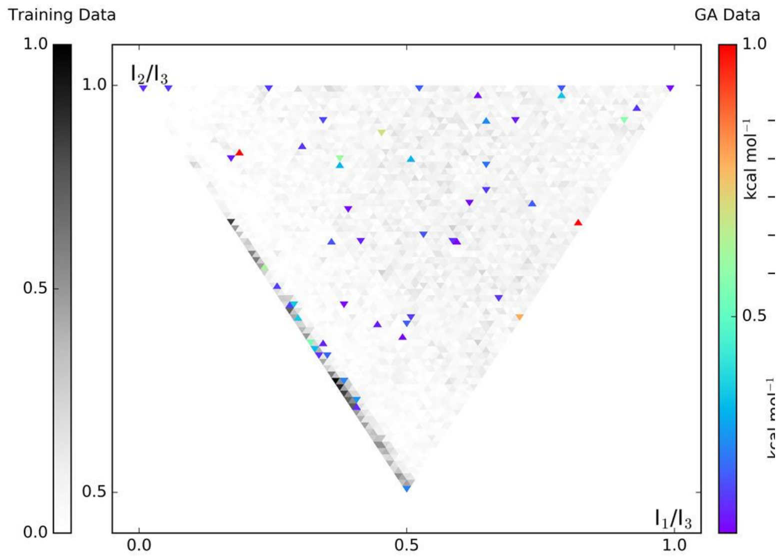
Computing K is $O(N^2)$
Solving KRR is $O(N^3)$!

New features with
each training point = $O(N)$

Reframing: *"Dial in performance vs accuracy tradeoff"*

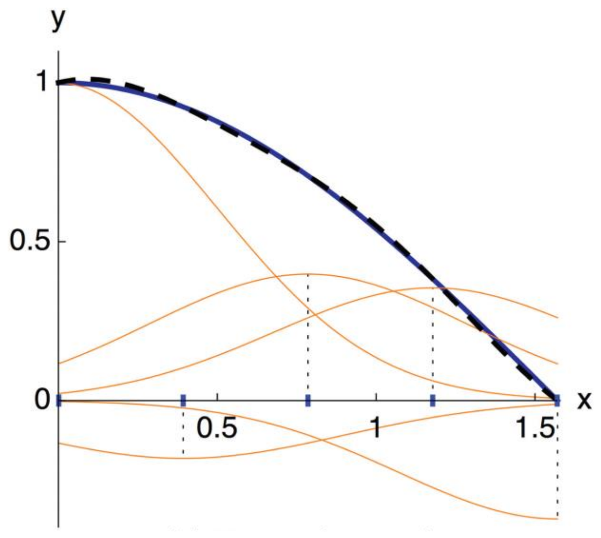
Some routes for addressing scaling issue

Pick "high value points"



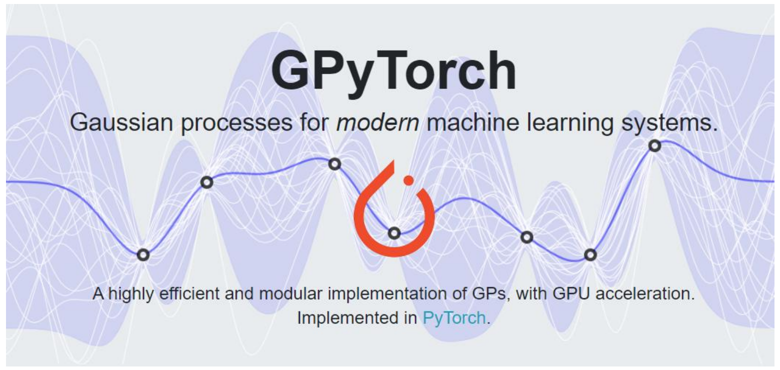
Ref: Browning et al. JPCL. (2017)

Adjust basis points



Ref: Snelson, Ghahramani. NuerIPS (2005)

Deploy on GPUs


GPyTorch
Gaussian processes for *modern* machine learning systems.
A highly efficient and modular implementation of GPs, with GPU acceleration.
Implemented in PyTorch.

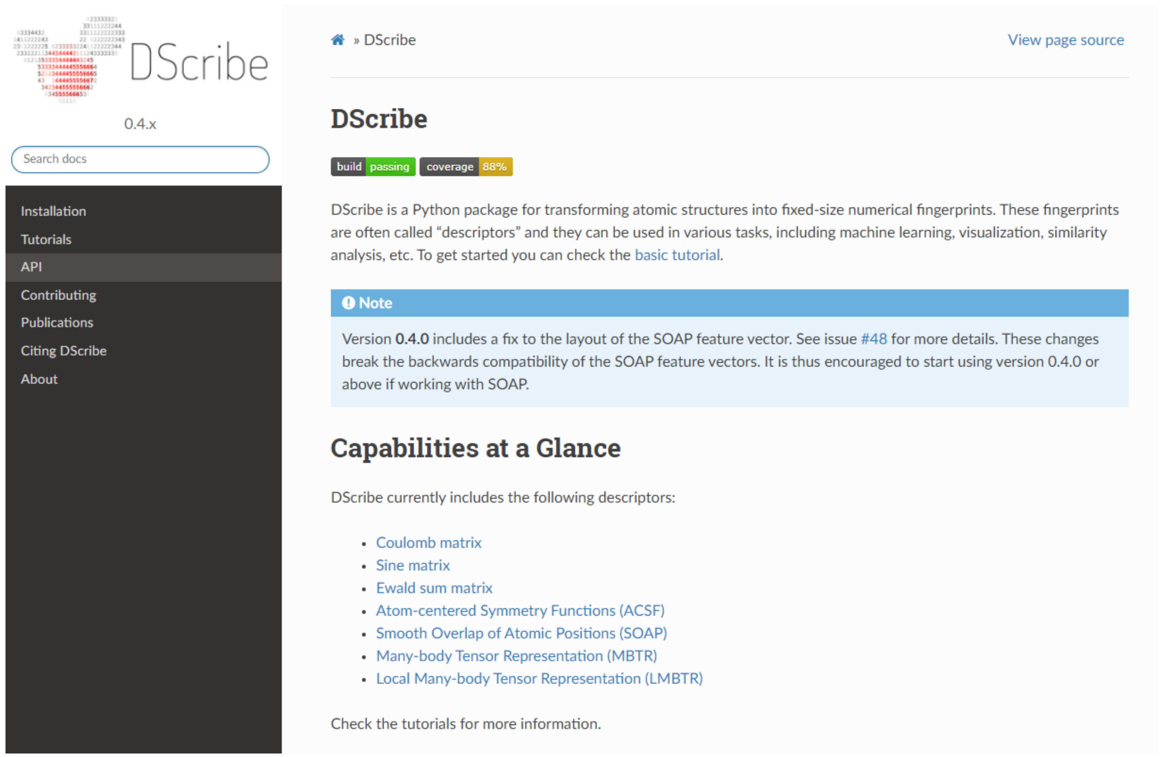
You can use GPR with large datasets, its just more work

A good reference: J. Emmanuel Johnson's Research Journal

Codes

DScribe

QML



DScribe

0.4.x

Search docs

Installation
Tutorials
API
Contributing
Publications
Citing DScribe
About

» DScribe

View page source

DScribe

build passing | coverage 88%

DScribe is a Python package for transforming atomic structures into fixed-size numerical fingerprints. These fingerprints are often called "descriptors" and they can be used in various tasks, including machine learning, visualization, similarity analysis, etc. To get started you can check the basic tutorial.

Note

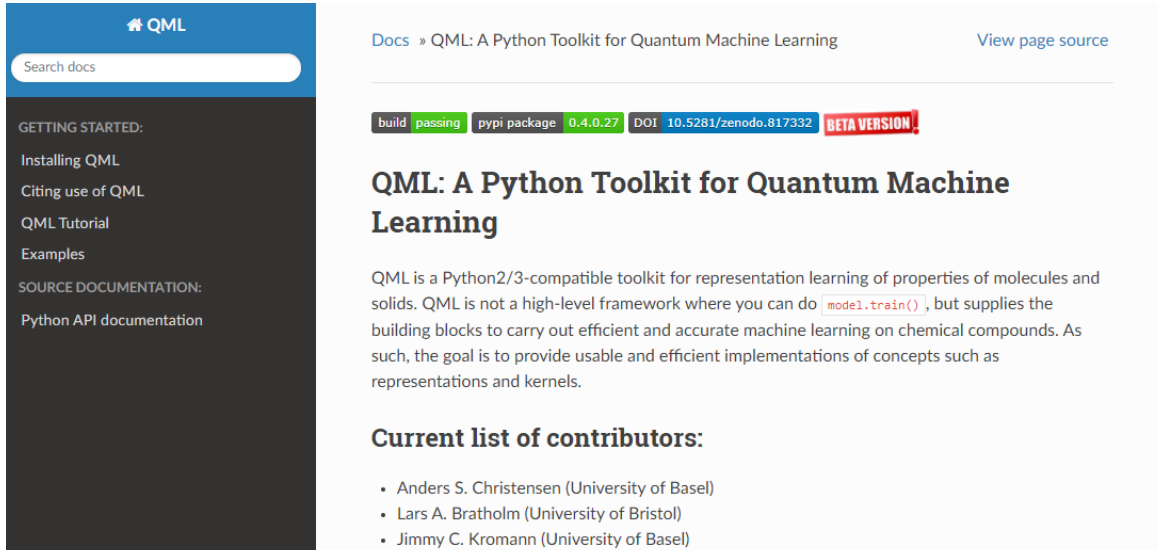
Version **0.4.0** includes a fix to the layout of the SOAP feature vector. See issue #48 for more details. These changes break the backwards compatibility of the SOAP feature vectors. It is thus encouraged to start using version 0.4.0 or above if working with SOAP.

Capabilities at a Glance

DScribe currently includes the following descriptors:

- Coulomb matrix
- Sine matrix
- Ewald sum matrix
- Atom-centered Symmetry Functions (ACSF)
- Smooth Overlap of Atomic Positions (SOAP)
- Many-body Tensor Representation (MBTR)
- Local Many-body Tensor Representation (LMBTR)

Check the tutorials for more information.



QML

Search docs

GETTING STARTED:
Installing QML
Citing use of QML
QML Tutorial
Examples
SOURCE DOCUMENTATION:
Python API documentation

Docs » QML: A Python Toolkit for Quantum Machine Learning

View page source

build passing | pypi package 0.4.0.27 | DOI 10.5281/zenodo.817332 | BETA VERSION

QML: A Python Toolkit for Quantum Machine Learning

QML is a Python2/3-compatible toolkit for representation learning of properties of molecules and solids. QML is not a high-level framework where you can do `model.train()`, but supplies the building blocks to carry out efficient and accurate machine learning on chemical compounds. As such, the goal is to provide usable and efficient implementations of concepts such as representations and kernels.

Current list of contributors:

- Anders S. Christensen (University of Basel)
- Lars A. Bratholm (University of Bristol)
- Jimmy C. Kromann (University of Basel)

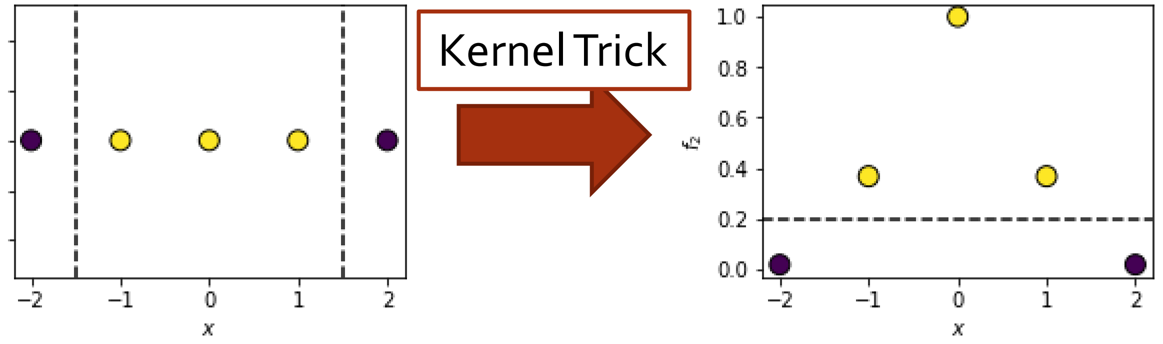
Conclusions and Outlook

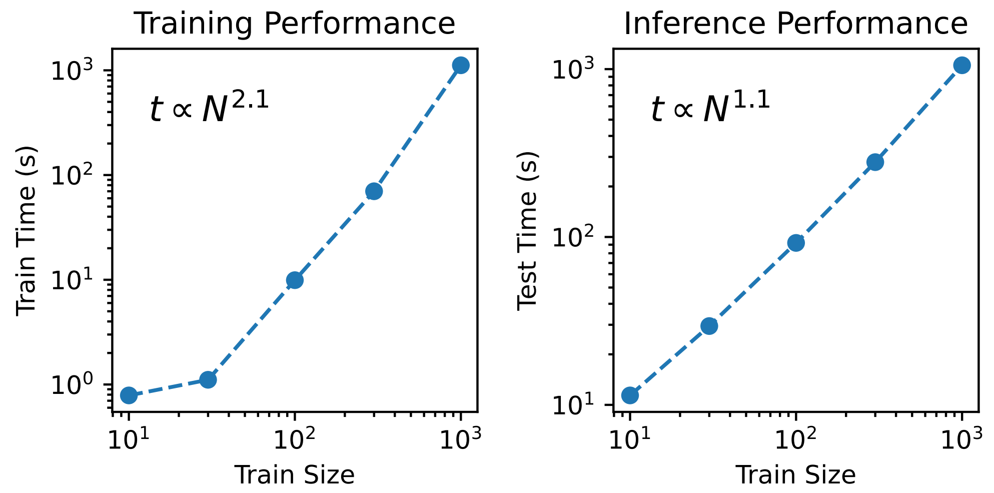
Key bits to understand:

1. Kernel methods simplify learning through *similarity functions*
2. There are many made for molecules
3. Learning curve to using them and other tradeoffs (e.g., scaling)

Not just "historical methods"

- Extremely good predictive accuracy
- Active area of research


Kernel Trick
x
x
f_2
−2
−1
0
1
2
0.0
0.2
0.4
0.6
0.8
1.0


Training Performance
$t \propto N^{2.1}$
Train Time (s)
Train Size
Inference Performance
$t \propto N^{1.1}$
Test Time (s)
Train Size
10^0
10^1
10^2
10^3