Practical Session: Regression and Classification

Jun Chen
Fujian Institute of Research on the Structure of Matter,
Chinese Academy of Sciences

July 14, 2025

Our goals today:

- Familiar with classification and regression
- Familiar with SVM/RF/xGB/NN methods
- Become an expert in GP/NN PES construction!

Let's form 10 teams (4-6 guys each team) for a machine learning challenge!

- **Project Requirements**

  Each team are recommended to complete at lease two tasks:

  - Regression/Classification: Choose at least one from Exercise 1 or 2.
  - PES Fitting: Complete GP or NN fitting from Exercise 3 or 4.

- **Showcase Your Work**

  Each team will have 3 minutes for a flash talk (3-5 slides) to present results.

- **Tools & Resources**

  - Recommended: Python (scikit-learn, TensorFlow, PyTorch, etc.).
  - Alternatives: MATLAB or Fortran are also recommended.
  - AI Assistance: Feel free to use AI tools (GPT, Gemini, etc.).

Exercise 1: predict the activation energy E_a → regression

Exercise 2: predict the activity of catalyst → classification

Exercise 3: potential energy surface using Gaussian Process

Exercise 4: potential energy surface using Neural Networks

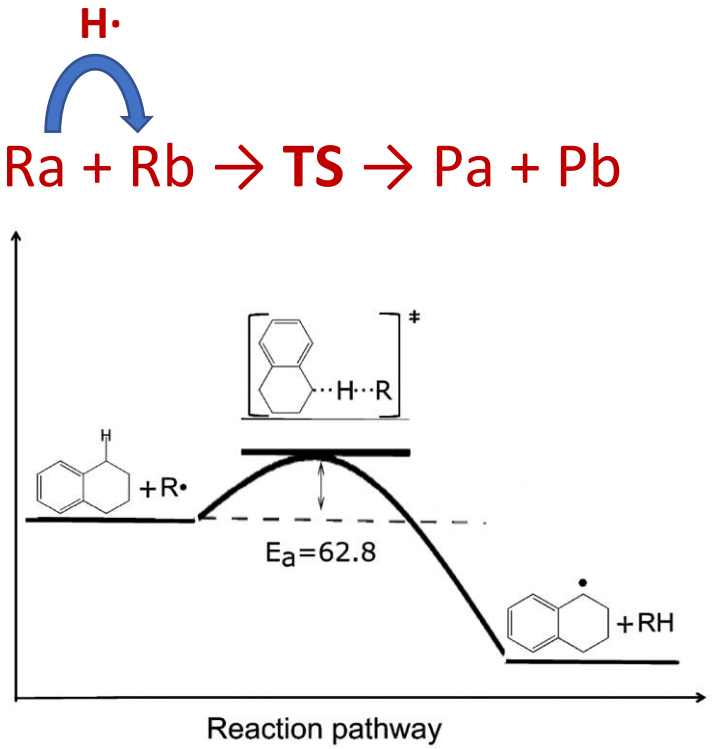
all materials can be found in:
google drive: MLAI for Molecular Science Summer 2025 / Day 1 / practical_section_7.14

or

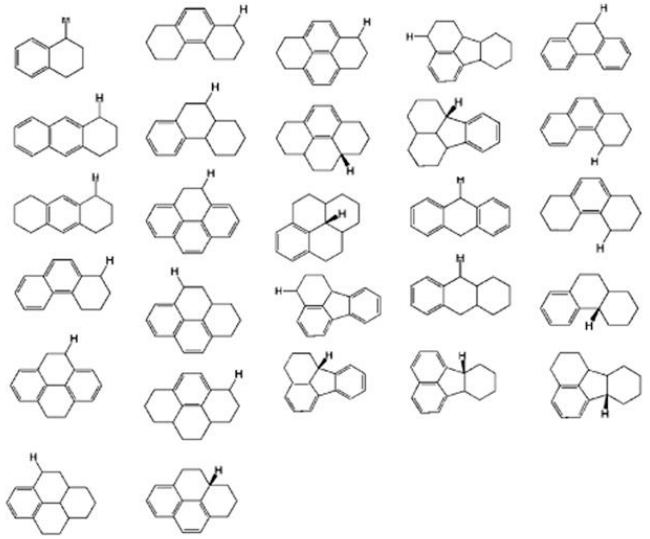
https://github.com/njuchenjun/nyu_summer_school_2025 → practical_section_7.14.tar

Exercise 1: predict the activation energy E_a → regression

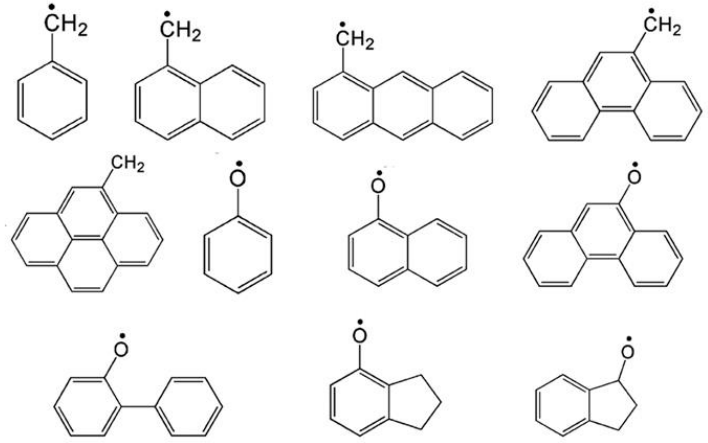
Exercise 1: predict the activation energy E_a → regression



the Ra set



the Rb set



task 1: prepare of dataset. Structure of Ra and Rb -> **RDkit** (SMILES, …) -> features (xlsx file)

task 2: predict E_a of the TS. Features of Ra and Rb -> ML models -> E_a labels

Exercise 1: predict the activation energy E_a → regression

descriptors type I:
physical chemistry knowledge

- **BDE_A**, **BDE_B**: Bond dissociation free energy
- **V_A**, **V_B**: Molecular volume
- **BertzCT_A**, **BertzCT_B**: Bertz complexity index, describing the molecule's topological complexity
- **κ_A**, **κ_B**: Molecular shape descriptors
- **NOcount**: Number of nitrogen (N) and oxygen (O) atoms
- **N_Ar_R**: Number of aromatic rings
- **N_Sat_C**: Number of saturated rings
- **H_K_α_A**, **H_K_α_B**: Hall–Kier α values, describing the molecule's topological polarity

descriptors type II:
microscopic structure

```python
def get_molecular_features(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    fp = AllChem.GetMorganFingerprintAsBitVect(mol, radius=2, nBits=128)
    fp_array = np.array(fp)
    return fp_array
```

```python
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Descriptors
```

Out:

-20.8591	-11.3542	-14.0035	17.61272	21.97954	3.397233
-32.6578	-18.1333	-13.4749	28.00932	22.2043	10.34528
-24.4438	-11.4466	-15.2457	18.08208	22.83708	5.564182
-22.1089	-36.3316	-30.9302	23.33021	35.98293	-5.28019
-23.8379	-13.3165	-18.1229	4.91371	26.03094	2.287888
-20.9099	-27.5391	12.35237	47.02663	72.26748	61.18955
-26.8044	-9.84174	1.944043	7.38704	7.95282	8.066493

```python
class AutoEncoder(nn.Module):
    def __init__(self):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 16),
            nn.ReLU(),
            nn.Linear(16, 6)
        )
        self.decoder = nn.Sequential(
            nn.Linear(6, 16),
            nn.ReLU(),
            nn.Linear(16, 64),
            nn.ReLU(),
            nn.Linear(64, 128)
        )

    def forward(self, x):
        code = self.encoder(x)
        reconstructed = self.decoder(code)
        return reconstructed
```

```
Epoch 5000, Loss: 0.0522
Epoch 10000, Loss: 0.0340
Epoch 15000, Loss: 0.0256
Epoch 20000, Loss: 0.0208
Epoch 25000, Loss: 0.0182
Epoch 30000, Loss: 0.0166
Epoch 35000, Loss: 0.0140
Epoch 40000, Loss: 0.0128
Epoch 45000, Loss: 0.0113
Epoch 50000, Loss: 0.0094
Epoch 55000, Loss: 0.0086
Epoch 60000, Loss: 0.0079
Epoch 65000, Loss: 0.0076
Epoch 70000, Loss: 0.0072
Epoch 75000, Loss: 0.0071
Epoch 80000, Loss: 0.0066
Epoch 85000, Loss: 0.0063
Epoch 90000, Loss: 0.0060
Epoch 95000, Loss: 0.0057
Epoch 100000, Loss: 0.0056
```

Morgan fingerprint of molecule

↓

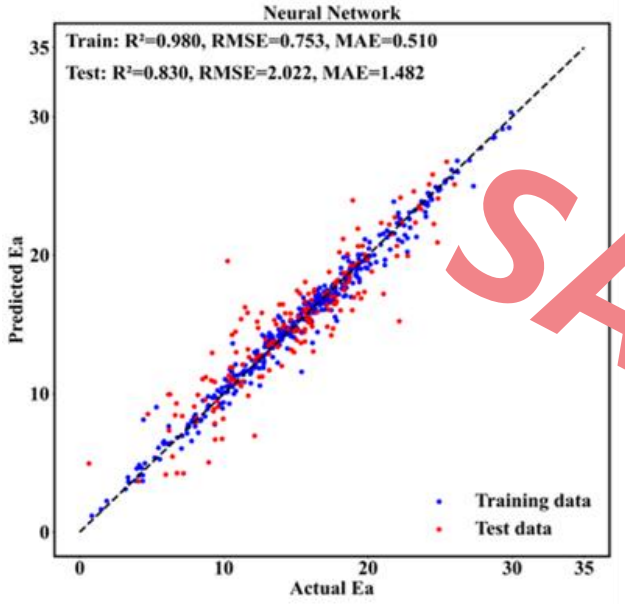
AutoEncoder 128D -> 6D

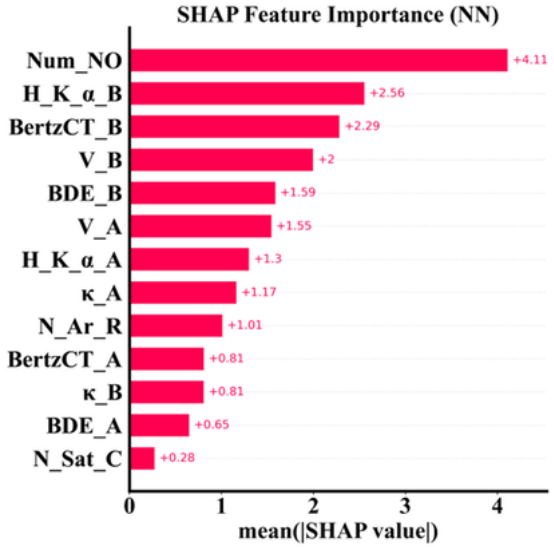
↓

Descriptor augmentation and training

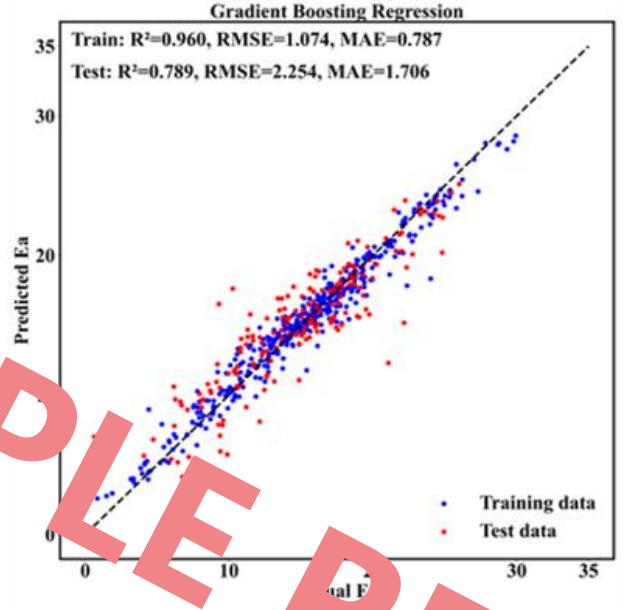
Exercise 1: predict the activation energy E_a → regression

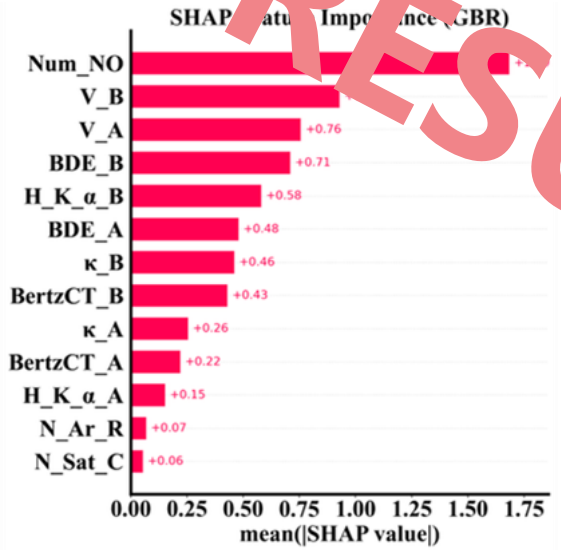
Neural Network



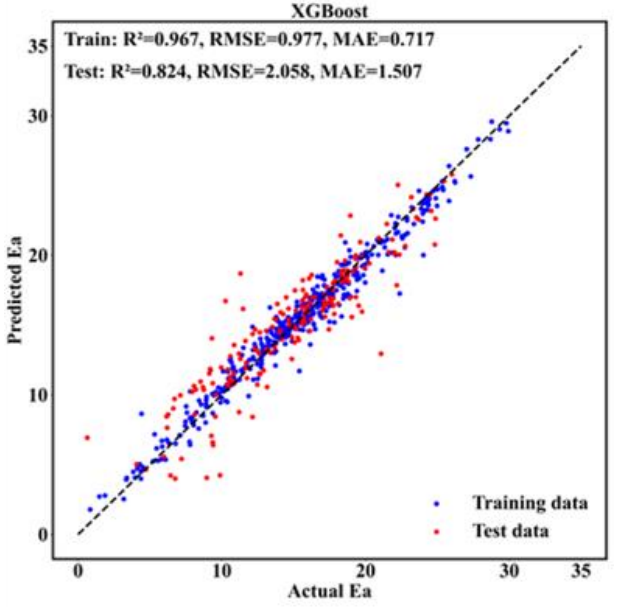


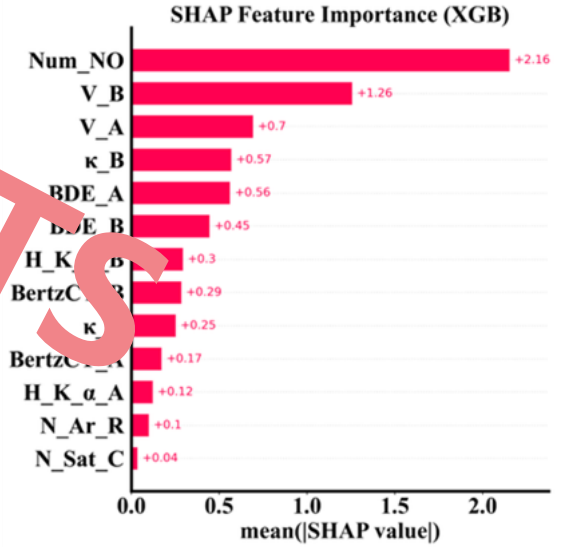
Gradient Boosting Regression





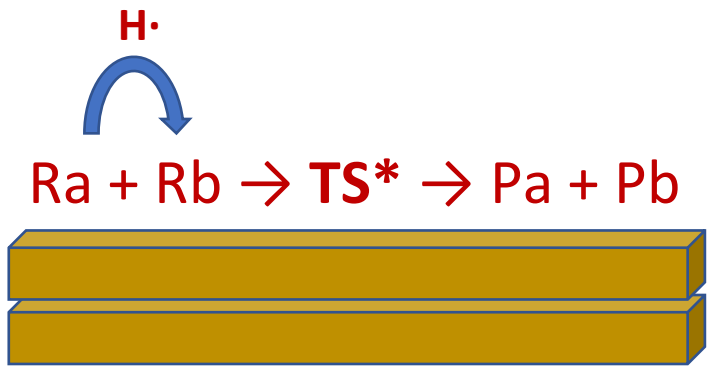
XGBoost



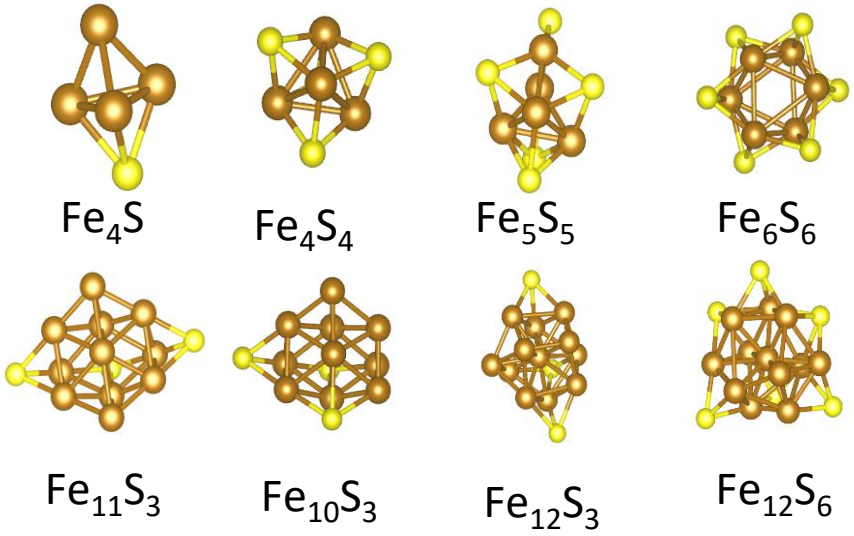


Exercise 2: predict the activity of catalyst → classification

Exercise 2: predict the activity of catalyst → classification



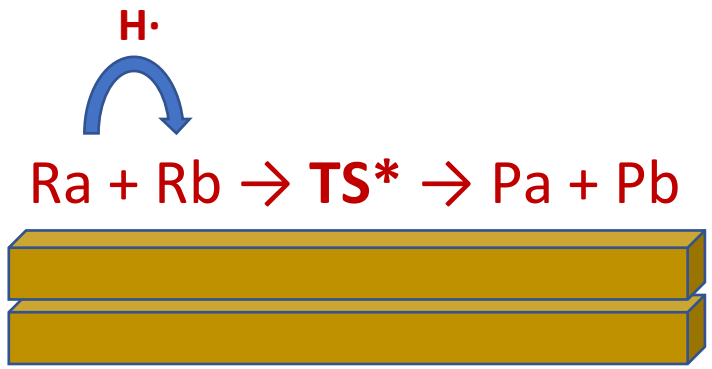
the catalyst set



Assuming the previous gas-phase reaction occurs on a metal cluster catalyst (heterogeneous catalysis model), can the transition state energy barrier be lowered?

Our goal: to find the optimal **Ra/Rb/catalyst** combination that facilitates the HAT (H atom transfer) reaction most easily.

Exercise 2: predict the activity of catalyst → classification



simplified as:

1. Ra → Pa + H·(ads)
2. Rb + H·(ads) → Pb
3. the structure of cluster

Directly calculating the energy barrier for the Ra+Rb reaction on a cluster is somewhat difficult, but we can assume the reaction energy barrier is determined by these factors:

1. The process of Ra losing a hydrogen atom to form Pa;
2. The process of Rb gaining a hydrogen atom to form Pb;
3. The electronic structure properties of the cluster.

task 1: Prepare of dataset. Structures -> **RDkit** (SMILES, …) -> features (xlsx file)

task 2: Make labels. TS of HAT processes (the elementary reaction).

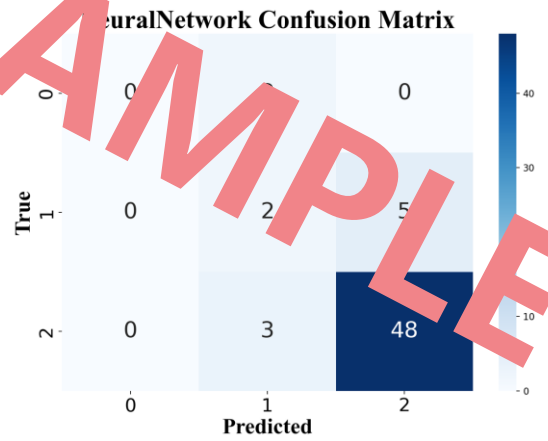
Exercise 2: predict the activity of catalyst → classification

Labeled as 3 classes:

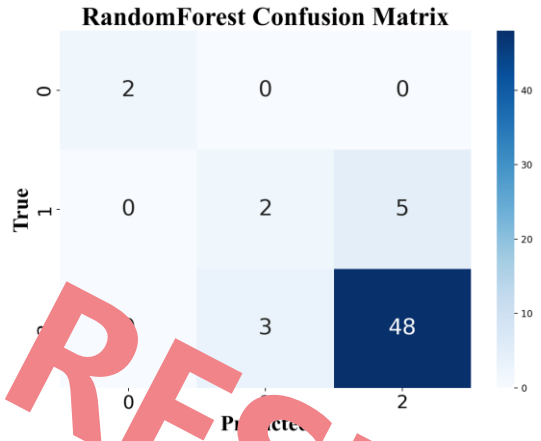
good:	**2**
medium:	**1**
fair:	**0**

rules:

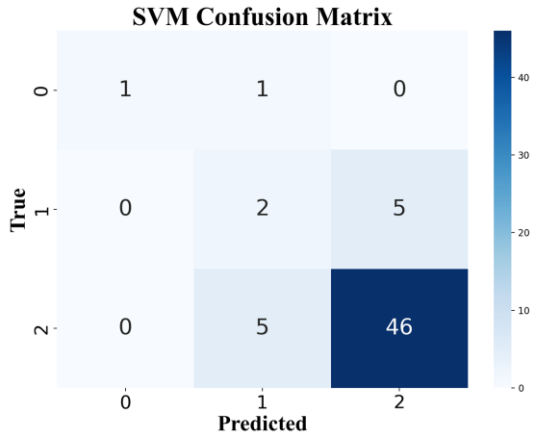
1. delta*H*< -0.5: +1
2. post*E*a< 3.2: +1



Accuracy
: 83%



Accuracy
: 87%



Accuracy
: 82%

Exercise 3: potential energy surface using Gaussian Process

Exercise 4: potential energy surface using Neural Networks

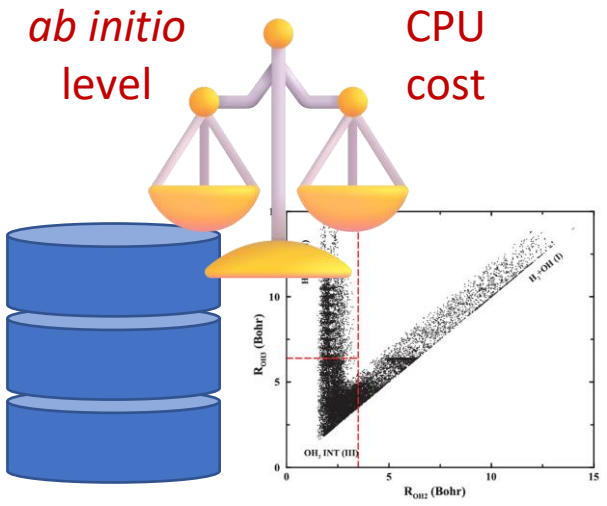
Exercise 3: potential energy surface using Gaussian Process

Exercise 4: potential energy surface using Neural Networks

the OH + H_2 -> H + H_2O reaction

1. dataset

16814 [structure, energy] points at CCSD(T)-f12a/AVTZ level



J. Chem. Phys., 2013, 138(15): 154301.

2. model

? choice

Shepard interpolate

PIP fitting

gaussian process

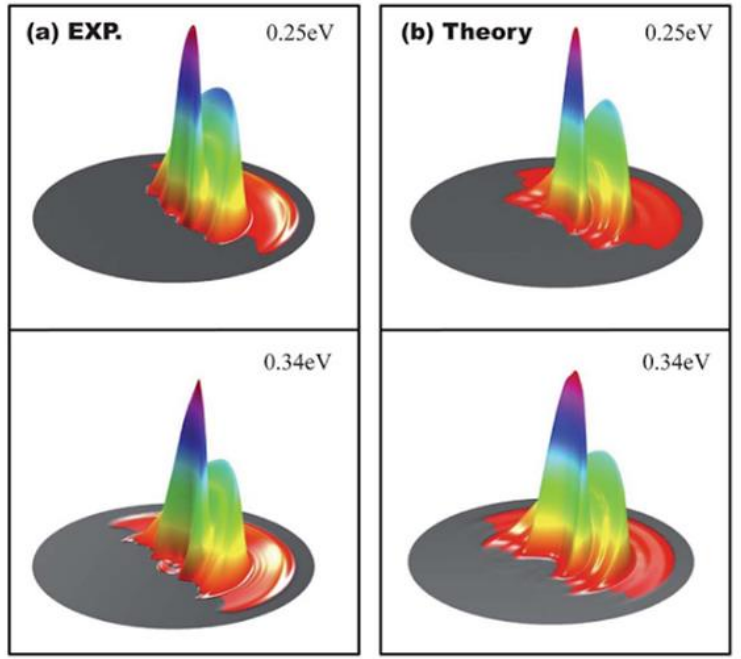
neural networks

? consider

fitting accuracy

evaluation speed

3. link to dynamics



Faraday Discussions, 2012, 157: 101-111.

Exercise 3: potential energy surface using Gaussian Process

Exercise 4: potential energy surface using Neural Networks

our goal(s):

1. error distribution plot

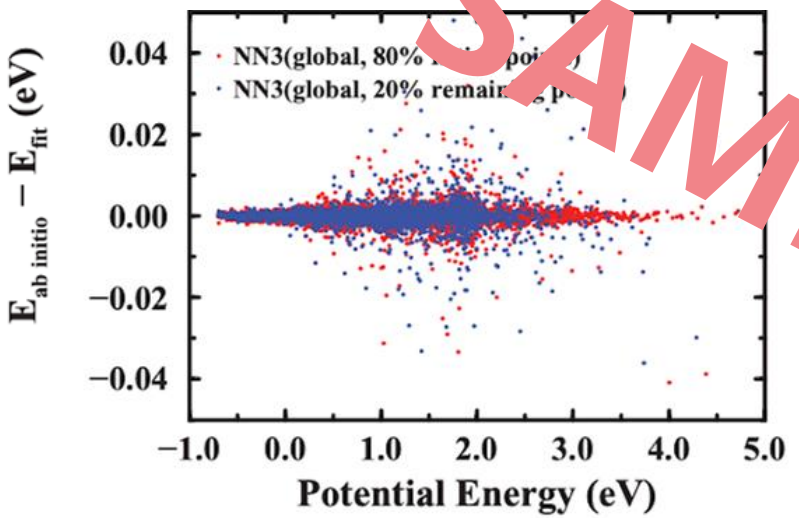
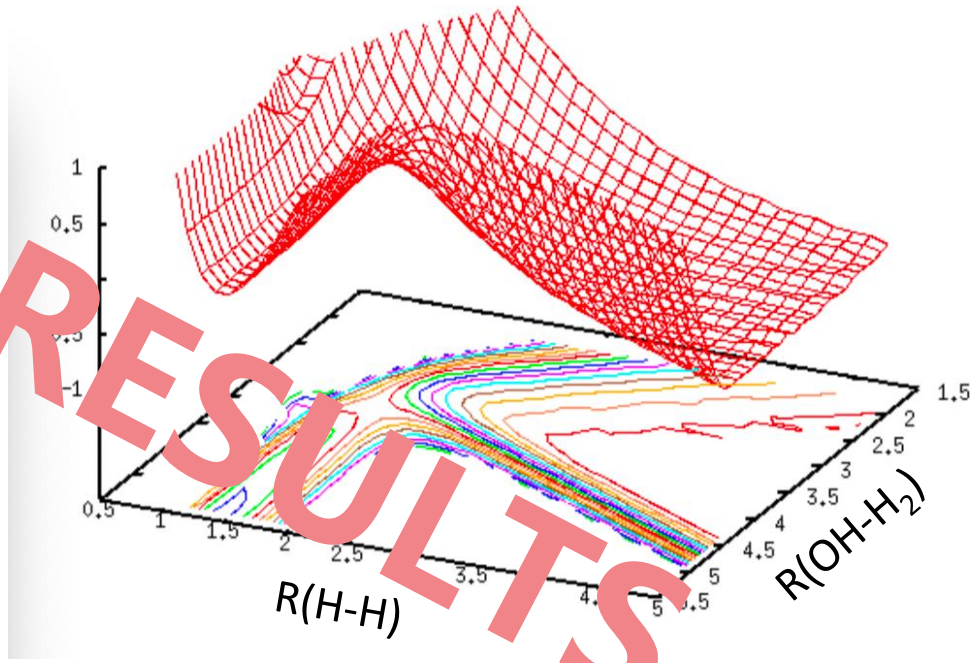


FIG. 2. The fitting errors for all the data points in NN1, NN2, and NN3 PESs, as a function of their corresponding *ab initio* energies with respect to OH + H_2.

J. Chem. Phys., 2013, 138(15): 154301.

2. 2D cut pf PES plot



Exercise 3: potential energy surface using Gaussian Process

A. the dataset: oh3-abinitio.txt

columns 1-6: distances / a.u.

O-H1, O-H2, O-H3, H1-H2, H1-H3, H2-H3

O-H1 <= O-H2 <= O-H3

column 7: potential energy / eV

relative to the OH + H_2 asymptotic valley

B. scikit-learn, GPflow, GPy, GPyTorch, George

C. a sample code: gp_training.m

use "MATLAB" and the "gpml" toolbox

do the fit and get a "GPinput.txt" file

use "fortran_gp_pes" to evaluate the PES

Discussions:

1. size of training set
2. distrib. of training set
3. the kernel:

covfunc, likfunc, ...

4. use R or 1/R

Exercise 3: potential energy surface using Gaussian Process

Future works:

1. **Permutation Invariance**

SOAP-GAP https://libatoms.github.io/GAP/

2. **construct small dataset for GP**

clustering, active learning

Exercise 4: potential energy surface using Neural Networks

A. the dataset: oh3-abinitio.txt

columns 1-6: distances / a.u.

O-H1, O-H2, O-H3, H1-H2, H1-H3, H2-H3

O-H1 <= O-H2 <= O-H3

column 7: potential energy / eV

relative to the OH + H_2 asymptotic valley

B. PyTorch or TensorFlow

C. a sample code: nn_training.tar

use "nn.f" and "FORTRAN" compiler

compile, then run fittings, and get "W??.txt" files

use "fortran_nn_pes" to evaluate the PES

Discussions:

1. training/validation set
2. neurons, hidden layers
3. the random number
4. use R, 1/R, or exp(-aR)
5. excitation function:
   tansig, logsig, ReLU

Exercise 4: potential energy surface using Neural Networks

Future works:

1. **Permutation Invariance**

   PIP-NN, FI-NN, GNN, atomic-NN

2. **compare with general potentials (MACE / DPA-3)**

3. **how the dataset constructed?**

   active learning

Thank You!

Prof. Dr. Jun Chen
Email: chenjun@fjirsm.ac.cn / njuchenjun@gmail.com
https://orcid.org/0000-0002-8021-7458
https://scholar.google.com/citations?user=zbCkt7gAAAAJ