PRINCETON UNIVERSITY

GEO421 Special Lecture

An Introductory Course on

Deep **P**otential **M**olecular **D**ynamics (**DeePMD**)

Yifan Li

Department of Chemistry, Princeton University

Dec 05, 2023

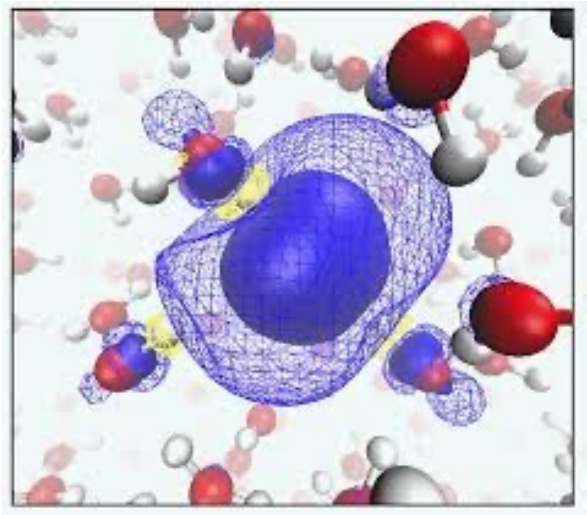
Lecture Outline

- Theoretical foundations of DeePMD
  - Recommended reading:

- A hands-on tutorial of using DeePMD-kit
  - Tutorial link: https://github.com/Yi-FanLi/GEO421-DPTutorial

Why Machine Learning Potential: Accuracy and Efficiency

Ab initio Molecular Dynamics



on-the-fly solution of the Kohn-Sham equations

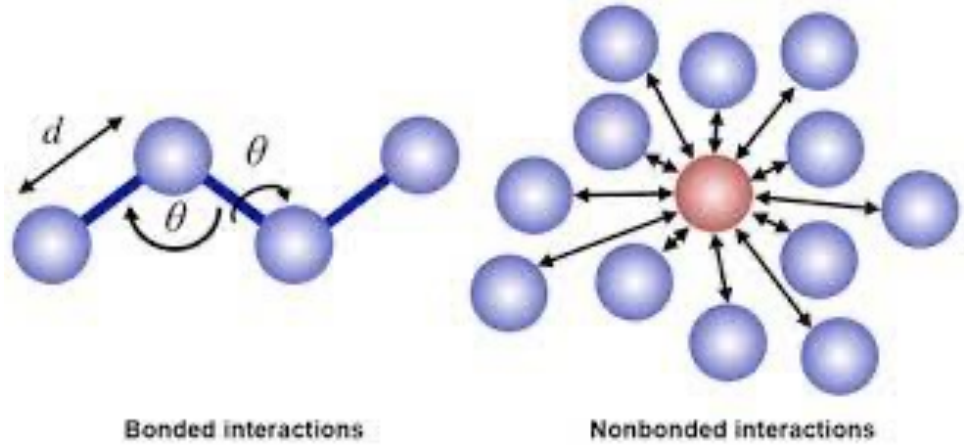
$$\left(-\tfrac{1}{2}\nabla^2 + v_s^{\mathrm{KS}}(\boldsymbol{r})\right)\psi_i(\boldsymbol{r}) = \varepsilon_i\psi_i(\boldsymbol{r})$$

👍 High accuracy

👎 High computational cost

limited to ~100s atoms, ~100s picosecond

Empirical Forcefield-based Molecular Dynamics



$$U\left(\mathbf{r}^N\right) = \sum_{\text{bonds}} \frac{k_i}{2}\left(l_i - l_{i,0}\right)^2 + \sum_{\text{angles}} \frac{k_i}{2}\left(\theta_i - \theta_{i,0}\right)^2$$
$$+ \sum_{\text{dihedrals}} \frac{V_n}{2}(1 + \cos(n\omega - \gamma))$$
$$+ \sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}}\right)$$

👍 High efficiency

👎 Poor transferability

Machine **L**earning **P**otential (**MLP**) combines the accuracy of aimd and the efficiency of empirical forcefield.

Machine Learning and Deep Neural Networks

- Machine learning: a branch of artificial intelligence that enables computers to learn and make decisions from data without being explicitly programmed.
- Supervised learning: a type of machine learning where an algorithm is trained on labeled data to learn the relationship between input variables and output labels.
- Deep neural network of n layers: $\mathcal{N} = \mathcal{L}^{(n)} \circ \mathcal{L}^{(n-1)} \circ \cdots \circ \mathcal{L}^{(1)}$
- Each layer is a function of the form:

"timestep" in ResNet, dimension N_2

activation function: N_2 to N_2 mapping

$$\boldsymbol{y} = \mathcal{L}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{b}) = \begin{cases} \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right) + \boldsymbol{x}, & \text{ResNet and } N_2 = N_1, \\ \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right) + \{\boldsymbol{x}, \boldsymbol{x}\}, & \text{ResNet and } N_2 = 2N_1, \\ \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right), & \text{otherwise,} \end{cases}$$

output, dimension N_2

input, dimension N_1

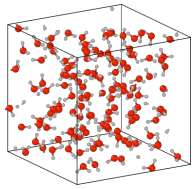
parameter, $N_1 \times N_2$ matrix

parameter, N_2 dimension

Workflow of Using DeePMD

Labeling

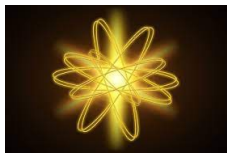
Input:



Atomic Coordinate

VASP CP2K QUANTUMESPRESSO

Label:



Energy and Force

Training

Labeled Data:
Atomic Coordinate
Energy and Force

DeePMD-kit

A Model File:
model.pb

Running MD

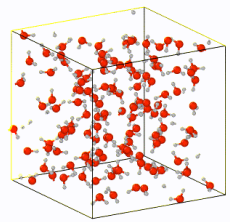
MD Input:
initial configuration
model file
MD settings

LAMMPS

FAST. FLEXIBLE. FREE. GROMACS

OpenMM

MD Trajectory:



Analysis

MD Trajectory

MDTRAJ MD ANALYSIS VMD Visual Molecular Dynamics

Desired Properties

Key Components of a Deep Potential Model

- Energy decomposition:

$$E = \sum_i E_i = \sum_i \mathcal{F}_0\left(\mathcal{D}^i\right)$$

- 2 neural networks

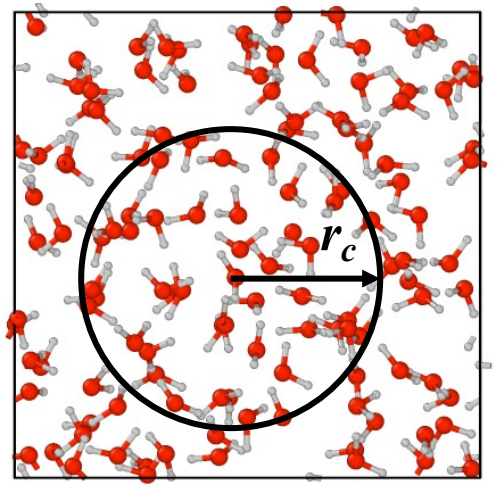
Coordinates R_i
Neighbors $\{R_j\}$ inside a cutoff r_c

→ **Embedding Network** →

Descriptor D^i

→ **Fitting Network** →

Atomic energy E_i



liquid water:
density ~ 1 g / cm^3

number of H_2O molecules in a r_c = 6 Å sphere: ~ 30

Each atoms has ~30 O and ~60 H atoms in its neighborhood.

Descriptor types:

- se_e2_a
- se_e2_r
- se_e3
- se_a_tpe
- hybrid

Atomic force:

$$F_{i,\alpha} = -\frac{\partial E}{\partial r_{i,\alpha}}$$

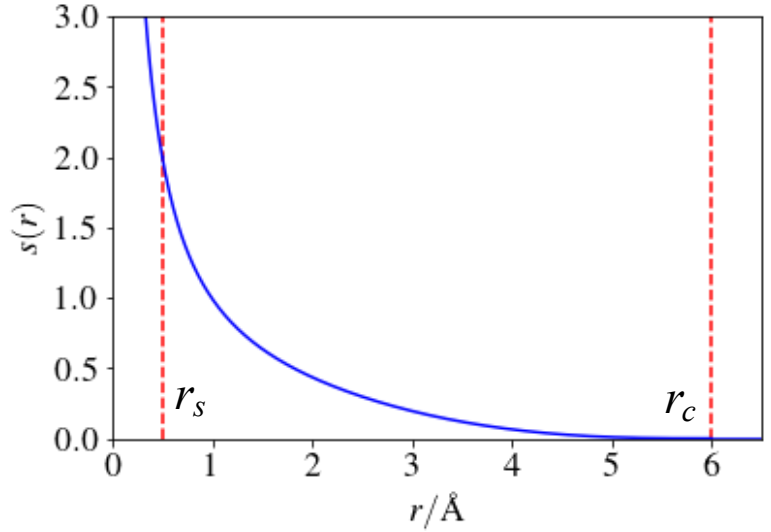
Virial tensor:

$$\Xi_{\alpha\beta} = -\sum_{\gamma} \frac{\partial E}{\partial h_{\gamma\alpha}} h_{\gamma\beta}$$

Descriptors se_e2_a and se_e2_r

- Switching function $s(r)$: $x = \frac{r - r_s}{r_c - r_s}$

$$s(r) = \begin{cases} \frac{1}{r}, & r < r_s, \\ \frac{1}{r}\left[x^3\left(-6x^2 + 15x - 10\right) + 1\right], & r_s \le r < r_c, \\ 0, & r \ge r_c, \end{cases}$$



- Find N_c neighbors of atom i (for H2O, 50 O and 100 H)

- Coordinate matrix R^i:

$$(\mathcal{R}^i)_j = \begin{cases} \left\{ s(r_{ij}) \quad \frac{s(r_{ij})x_{ij}}{r_{ij}} \quad \frac{s(r_{ij})y_{ij}}{r_{ij}} \quad \frac{s(r_{ij})z_{ij}}{r_{ij}} \right\}, & \text{se\_e2\_a} \\ \{s(r_{ij})\}, & \text{se\_e2\_r} \end{cases}$$

R^i: $N_c \times 4$ or $N_c \times 1$

$$N_c \begin{pmatrix} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i i}) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{pmatrix}$$

- Embedding network: $(\mathcal{G}^i)_j = \mathcal{N}_{e,2}(s(r_{ij}))$

vector, dimension M — scalar

Descriptors se_e2_a and se_e2_r

- Embedding network maps a scalar to a vector of dimension M

$$(\mathcal{G}^i)_j = \mathcal{N}_{e,2}(s(r_{ij}))$$

vector, dimension M — scalar

- Embedding matrix:

$$\mathcal{G}^i = \begin{pmatrix} \mathcal{G}^i_{11} & \cdots & \mathcal{G}^i_{1M_<} & \mathcal{G}^i_{1M_<+1} & \cdots & \mathcal{G}^i_{1M} \\ \mathcal{G}^i_{21} & \cdots & \mathcal{G}^i_{2M_<} & \mathcal{G}^i_{2M_<+1} & \cdots & \mathcal{G}^i_{2M} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \mathcal{G}^i_{N_i 1} & \cdots & \mathcal{G}^i_{N_i M_<} & \mathcal{G}^i_{N_i M_<+1} & \cdots & \mathcal{G}^i_{N_i M} \end{pmatrix}$$

$N_c \times M$

$\mathcal{G}^i_<$ $N_c \times M_<$

- Descriptors D^i:

$$\mathcal{D}^i = \begin{cases} \frac{1}{N_c^2} (\mathcal{G}^i)^T \mathcal{R}^i (\mathcal{R}^i)^T \mathcal{G}^i_< & \text{se\_e2\_a} \\ \frac{1}{N_c} \sum_j (\mathcal{G}^i)_{jk} & \text{se\_e2\_r} \end{cases}$$

R^i: $N_c \times 4$ or $N_c \times 1$

$$N_c \left(\begin{array}{c|ccc} s(r_{1i}) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{ji}) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s(r_{N_i i}) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{array} \right)$$

Descriptor se_e3

- Descriptor se_e3 uses three-body information
- Embedding network of se_e3:

$$\left(\mathcal{G}^i\right)_{jk} = \mathcal{N}_{e,3}\left(\left(\theta_i\right)_{jk}\right) \qquad \left(\theta_i\right)_{jk} = \left(\mathcal{R}^i\right)_j \cdot \left(\mathcal{R}^i\right)_k$$

vector, dimension M — scalar

R^i: $N_c \times 4$ or $N_c \times 1$

$$N_c \left\{ \left(\begin{array}{c|ccc} s\left(r_{1i}\right) & \hat{x}_{1i} & \hat{y}_{1i} & \hat{z}_{1i} \\ \cdots & \cdots & \cdots & \cdots \\ s\left(r_{ji}\right) & \hat{x}_{ji} & \hat{y}_{ji} & \hat{z}_{ji} \\ \cdots & \cdots & \cdots & \cdots \\ s\left(r_{N_i i}\right) & \hat{x}_{N_i i} & \hat{y}_{N_i i} & \hat{z}_{N_i i} \end{array} \right) \right.$$

- Embedding tensor: $\mathcal{G}^i \in \mathbb{R}^{N_c \times N_c \times M}$
- Descriptor through tensor contraction: $\mathcal{R}^i \left(\mathcal{R}^i\right)^T \in \mathbb{R}^{N_c \times N_c}$

$$\mathcal{D}^i = \frac{1}{N_c^2}\left(\mathcal{R}^i \left(\mathcal{R}^i\right)^T\right) : \mathcal{G}^i$$

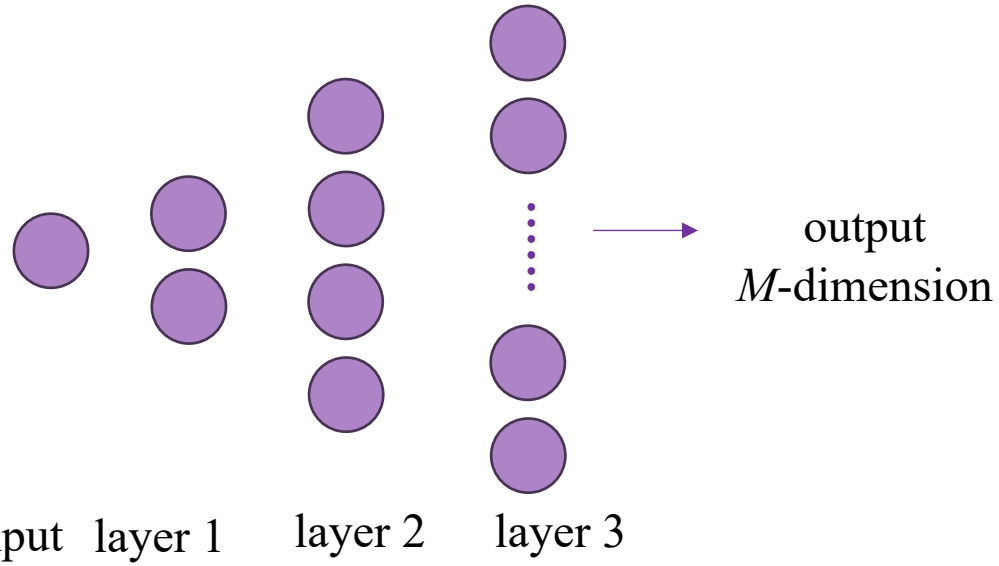
- D^i is a vector of dimension M.

Embedding Network and Fitting Network

- Embedding network:
  - Input: switching function $s(r_{ij})$
  - Output: embedding matrix $(\mathcal{G}^i)_j = \mathcal{N}_{e,2}(s(r_{ij}))$
  - 1-dimenion ⟶ M-dimension

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b}) = \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T\boldsymbol{w} + \boldsymbol{b}\right) + \{\boldsymbol{x},\boldsymbol{x}\},$$

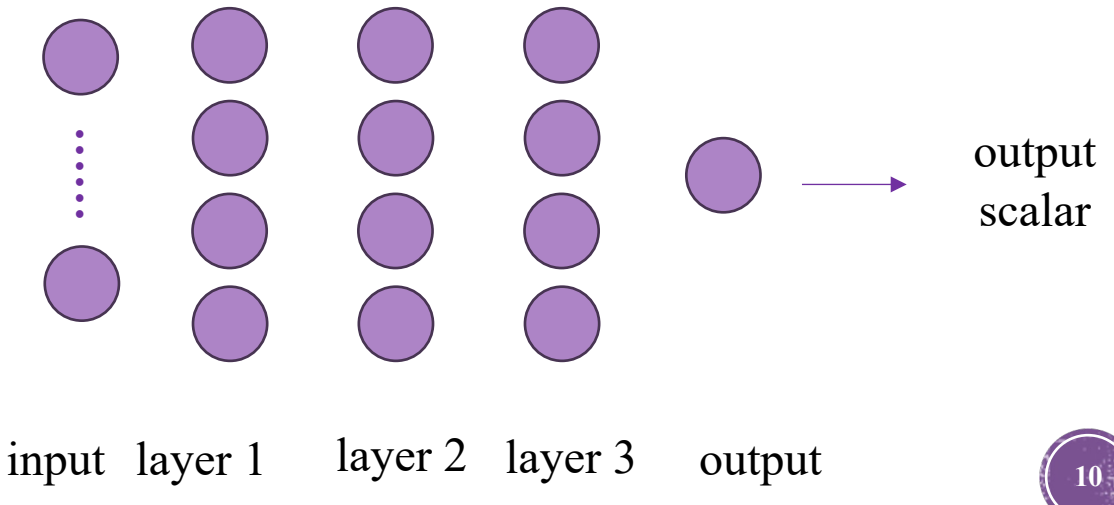
ResNet and $N_2 = 2N_1$



- Fitting network:
  - Input: descriptor D^i
  - Output: atomic energy E^i
  - $M \times M_<$ ⟶ scalar

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b}) = \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T\boldsymbol{w} + \boldsymbol{b}\right) + \boldsymbol{x},$$

ResNet and $N_2 = N_1$



Loss Function and Training Process

- Loss function

$$L(\boldsymbol{x};\boldsymbol{\theta},\tau)=\frac{1}{\mathcal{B}}\sum_{k\in\mathcal{B}}\sum_{\eta}p_{\eta}(\tau)L_{\eta}\left(\boldsymbol{x}^{k};\boldsymbol{\theta}\right)$$

- Contributions from different labels:

$$L_E(\boldsymbol{x};\boldsymbol{\theta})=\frac{1}{N}\left(E(\boldsymbol{x};\boldsymbol{\theta})-E^*\right)^2,$$

$$L_F(\boldsymbol{x};\boldsymbol{\theta})=\frac{1}{3N}\sum_{k=1}^{N}\sum_{\alpha=1}^{3}\left(F_{k,\alpha}(\boldsymbol{x};\boldsymbol{\theta})-F^*_{k,\alpha}\right)^2,$$

$$L_{\Xi}(\boldsymbol{x};\boldsymbol{\theta})=\frac{1}{9N}\sum_{\alpha,\beta=1}^{3}\left(\Xi_{\alpha\beta}(\boldsymbol{x};\boldsymbol{\theta})-\Xi^*_{\alpha\beta}\right)^2,$$

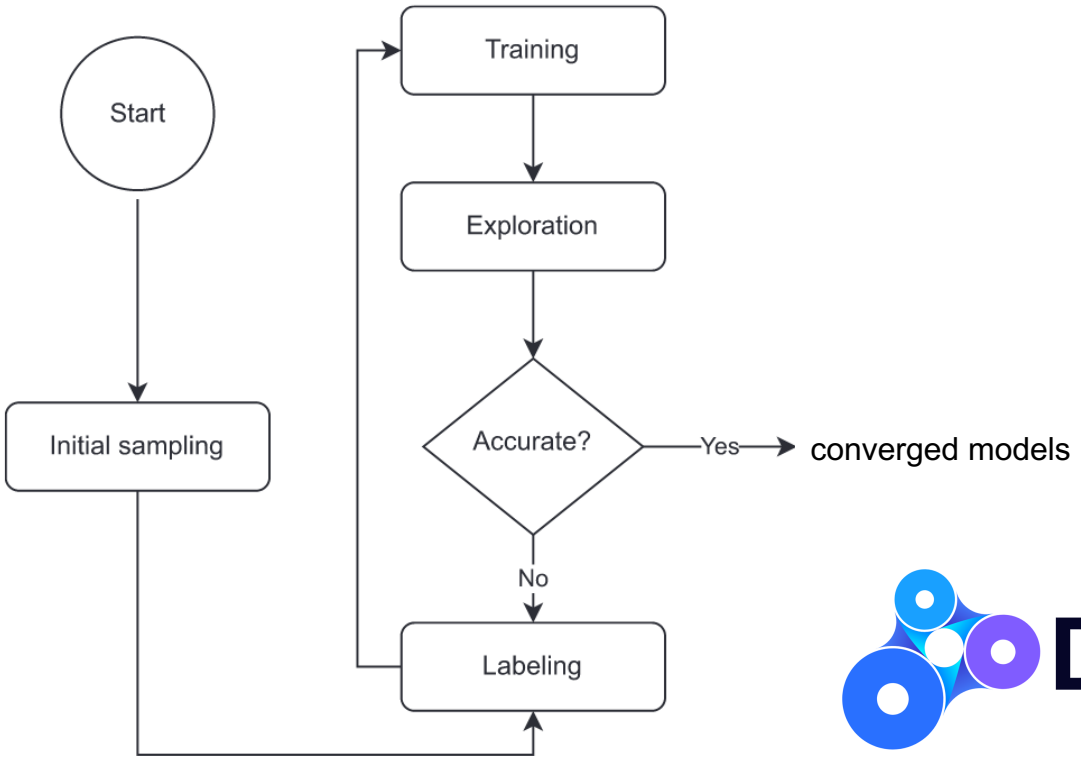
- Minimize L using stochastic gradient descent algorithm Adam

$$\boldsymbol{\theta}^*=\arg\min_{\boldsymbol{\theta}}\lim_{\tau\to+\infty}L(\boldsymbol{x};\boldsymbol{\theta},\tau)$$

- Exponential decay of learning rate: $\gamma(\tau)=\gamma^0 r^{\lfloor\tau/s\rfloor}$ $\quad r=\left(\frac{\gamma^{\text{stop}}}{\gamma^0}\right)^{\frac{s}{\tau^{\text{stop}}}}$

Concurrent Learning and DP-GEN

- **Several models** trained on a complete dataset give **consistent** predictions.
- Concurrent learning procedure



- Model deviation of a configuration:

$$\epsilon_{\boldsymbol{F},i}(\boldsymbol{x}) = \sqrt{\left\langle \left\| \boldsymbol{F}_i\left(\boldsymbol{x};\boldsymbol{\theta}_k\right) - \left\langle \boldsymbol{F}_i\left(\boldsymbol{x};\boldsymbol{\theta}_k\right)\right\rangle \right\|^2 \right\rangle}$$

- The configuration is accurate if:

$$\max_i \epsilon_{\boldsymbol{F},i}(\boldsymbol{x}) < \text{threshold}$$

- DP-GEN: a software to automatically perform concurrent learning

DP-GEN

Each iteration includes:

1. training 4 models
2. running MD and collect candidate configurations
3. doing DFT calculations

An Example Input File

- 4 parts: model, learning_rate, loss, training

```
{
    "model": {
        "type_map": [
            "O",
            "H"
        ],
        "descriptor": {
            "type": "se_a",
            "sel": [
                50,
                100
            ],
            "rcut_smth": 0.5,
            "rcut": 6.0,
            "neuron": [
                25,
                50,
                100
            ],
            "resnet_dt": false,
            "axis_neuron": 12,
            "seed": 3721519026
        },
        "fitting_net": {
            "neuron": [
                240,
                240,
                240
            ],
            "resnet_dt": true,
            "seed": 1632629719
        }
    },
```

$N_c = 50$ and 100

$r_s = 0.5$

$r_c = 6.0$

$M = 100$

$M_< = 12$

trainable parameters

$$\boldsymbol{y} = \mathcal{L}(\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{b}) = \begin{cases} \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right) + \boldsymbol{x}, & \text{ResNet and } N_2 = N_1, \\ \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right) + \{\boldsymbol{x}, \boldsymbol{x}\}, & \text{ResNet and } N_2 = 2N_1, \\ \hat{\boldsymbol{w}} \odot \boldsymbol{\phi}\left(\boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{b}\right), & \text{otherwise}, \end{cases}$$

$(1, 1, \ldots, 1)$

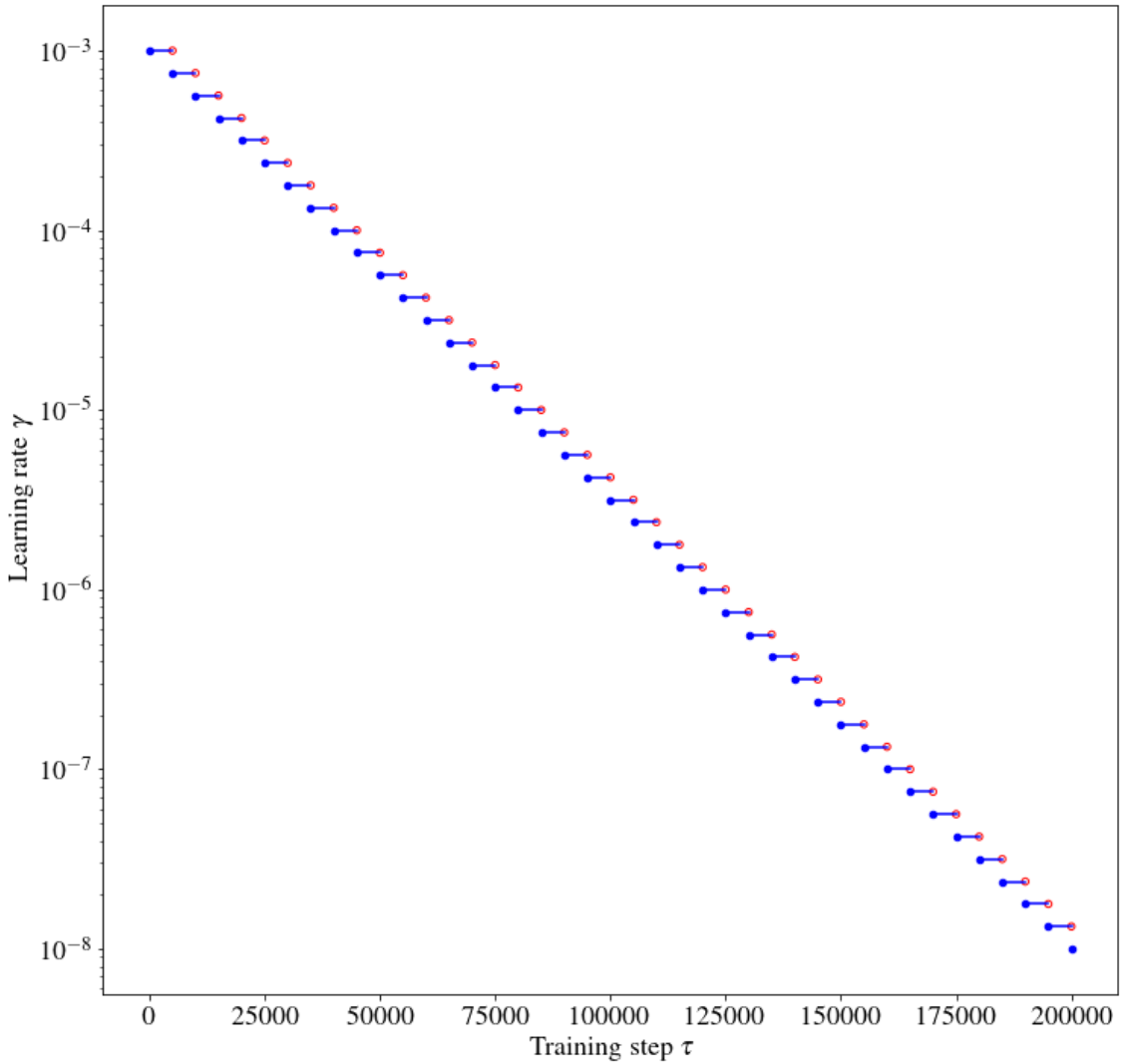
An Example Input File

```
34          "learning_rate": {
35              "type": "exp",
36              "start_lr": 0.001,
37              "stop_lr": 1e-08,
38              "decay_steps": 5000
39          },
40          "loss": {
41              "start_pref_e": 0.02,
42              "limit_pref_e": 2,
43              "start_pref_f": 1000,
44              "limit_pref_f": 1,
45              "start_pref_v": 0.0,
46              "limit_pref_v": 0.0
47          },
```

Exponential evolution of the prefactors of loss function:

$$p_\eta(\tau) = p_\eta^{\text{limit}} \left(1 - \frac{\gamma(\tau)}{\gamma^0}\right) + p_\eta^{\text{start}} \frac{\gamma(\tau)}{\gamma^0}$$

Exponential decay of the learning rate



An Example Input File

```
48        "training": {
49            "_set_prefix": "set",
50            "stop_batch": 200000,
51            "_batch_size": 1,
52            "disp_file": "lcurve.out",
53            "disp_freq": 100,
54            "save_freq": 10000,
55            "save_ckpt": "model.ckpt",
56            "disp_training": true,
57            "time_training": true,
58            "profiling": false,
59            "profiling_file": "timeline.json",
60            "_comment": "that's all",
```

```
61            "training_data": {
62                "systems": [
63                    "./data/train"
64                ],
65                "batch_size": [
66                    1
67                ]
68            },
69            "validation_data": {
70                "systems": [
71                    "./data/test"
72                ],
73                "batch_size": [
74                    1
75                ]
76            },
77            "seed": 1947382419
78        }
79    }
```

check the document for detailed explanations:

https://docs.deepmodeling.com/projects/deepmd/en/master/index.html

An Example Dataset

GEO421-DPTutorial / train / data / **train** /

Yi-FanLi upload training

Name
..
set.000
set.001
set.002
set.003
set.004
set.005
set.006
type.raw
type_map.raw

GEO421-DPTutorial / train / data / **test** /

Yi-FanLi upload training

Name
..
set.000
set.001
type.raw
type_map.raw

0, 0, 0, …, 1, 1, 1, …

0 for O
1 for H

- 3500 training data
- 927 validation data
- Each data point:
  - input: box and coord
  - label: energy and force

box.npy
coord.npy
energy.npy
force.npy

Questions?

Thanks for your attention!

Let’s do the hands-on tutorial!

https://github.com/Yi-FanLi/GEO421-DPTutorial