Modeling the interatomic potential by deep learning

Institute of applied physics and computational mathematics

Han Wang (王涵)

Introduction: what is molecular dynamics simulation

In mathematical language

1 准备初条件 $x = (q_1, q_2, \ldots q_N, p_1, p_2, \ldots, p_N)$

2 MD 模拟 $\dot{q}_i = \dfrac{\partial \mathcal{H}}{\partial p_i}$ $\dot{p}_i = -\dfrac{\partial \mathcal{H}}{\partial q_i}$

$$\mathcal{H} = \underbrace{\sum_{i=1}^{N} \frac{p_i^2}{2m_i}}_{\text{动能}} + \underbrace{E(q_1, \ldots, q_N)}_{\text{势能}}$$

3 数值求解得到“轨道” $\{q_i(t),\, p_i(t)\}$

Introduction: what is molecular dynamics simulation

In mathematical language

4 结果分析：物理可观测量可以写为坐标和动量 $\{q_i(t), p_i(t)\}$ 的函数

例如压力张量 $P = \frac{1}{T}\int_0^T \frac{1}{V}\sum_i \left(\frac{p_i(t)p_i(t)}{m_i} + r_i(t)F_i(t)\right)dt$

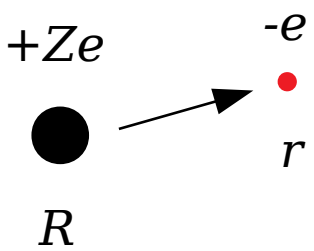
例如温度、密度、径向分布函数、扩散系数、粘性系数等

Introduction: the interatomic potential

$$E = E(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N)$$

Modeling and computation of potential is crucial for MD

Wave function as a description of electron



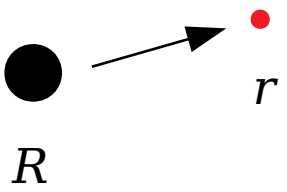
$$\Psi(\{r\};\{R\})$$

$$\rho(\{r\};\{R\}) = |\Psi(\{r\};\{R\})|^2$$

First principle: Schrodinger equation

$$\hat{H}(\{r\};\{R\})\Psi(\{r\};\{R\}) = E(\{R\})\,\Psi(\{r\};\{R\})$$

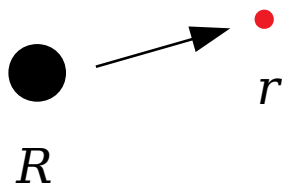
$$\hat{H} = -\sum_i \frac{\hbar^2}{2m_i}\nabla_i^2 - \sum_{i,I}\frac{Z_I}{|r_i - R_I|} + \sum_{i<j}\frac{1}{|r_i - r_j|}$$



First principle: Schrodinger equation

$$\hat{H}(\{r\};\{R\})\Psi(\{r\};\{R\}) = E(\{R\})\,\Psi(\{r\};\{R\})$$

$$\hat{H} = -\sum_i \frac{\hbar^2}{2m_i}\nabla_i^2 - \sum_{i,I}\frac{Z_I}{|r_i - R_I|} + \sum_{i<j}\frac{1}{|r_i - r_j|}$$



"... The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble. It therefore becomes desirable that approximate practical methods of applying quantum mechanics should be developed ..." [1]

[1] P. A. M. Dirac (1929), Proc. R. Soc. A, 123 (792): 714–733

The density functional theory

$$\hat{H}_{\mathrm{KS}}(r,\rho;\{R\})\psi(r;\{R\}) = E_{\mathrm{KS}}(\{R\})\psi(r;\{R\})$$

$$\hat{H}_{\mathrm{KS}} = -\frac{\hbar^2}{2m}\nabla^2 - \sum_I \frac{Z_I}{|r-R_I|} + \int \frac{\rho(r')}{|r-r'|}dr' + V_{\mathrm{xc}}[\rho](r)$$

$$\rho(r;\{R\}) = \sum_n f_n |\psi_n(r,\{R\})|^2$$

The density functional theory

$$\hat{H}_{\mathrm{KS}}(r,\rho;\{R\})\psi(r;\{R\}) = E_{\mathrm{KS}}(\{R\})\psi(r;\{R\})$$

$$\hat{H}_{\mathrm{KS}} = -\frac{\hbar^2}{2m}\nabla^2 - \sum_I \frac{Z_I}{|r-R_I|} + \int \frac{\rho(r')}{|r-r'|}dr' + V_{\mathrm{xc}}[\rho](r)$$

$$\rho(r;\{R\}) = \sum_n f_n |\psi_n(r,\{R\})|^2$$

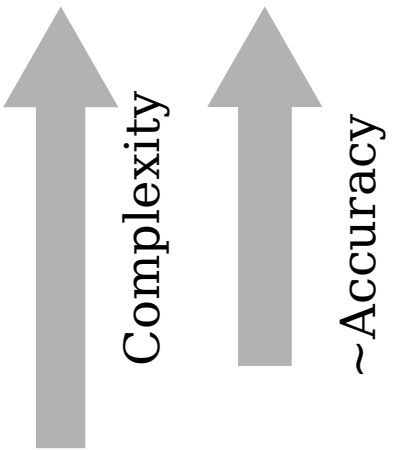
3 维非线性特征值问题：自洽求解。

$$\rho^{\mathrm{in}} \to \hat{H}_{\mathrm{KS}} \to \{\psi_n\} \to \rho^{\mathrm{out}} \to \text{a new } \rho^{\mathrm{in}} \to \dots$$

Other ab initio methods

Examples of “approximate practical methods”

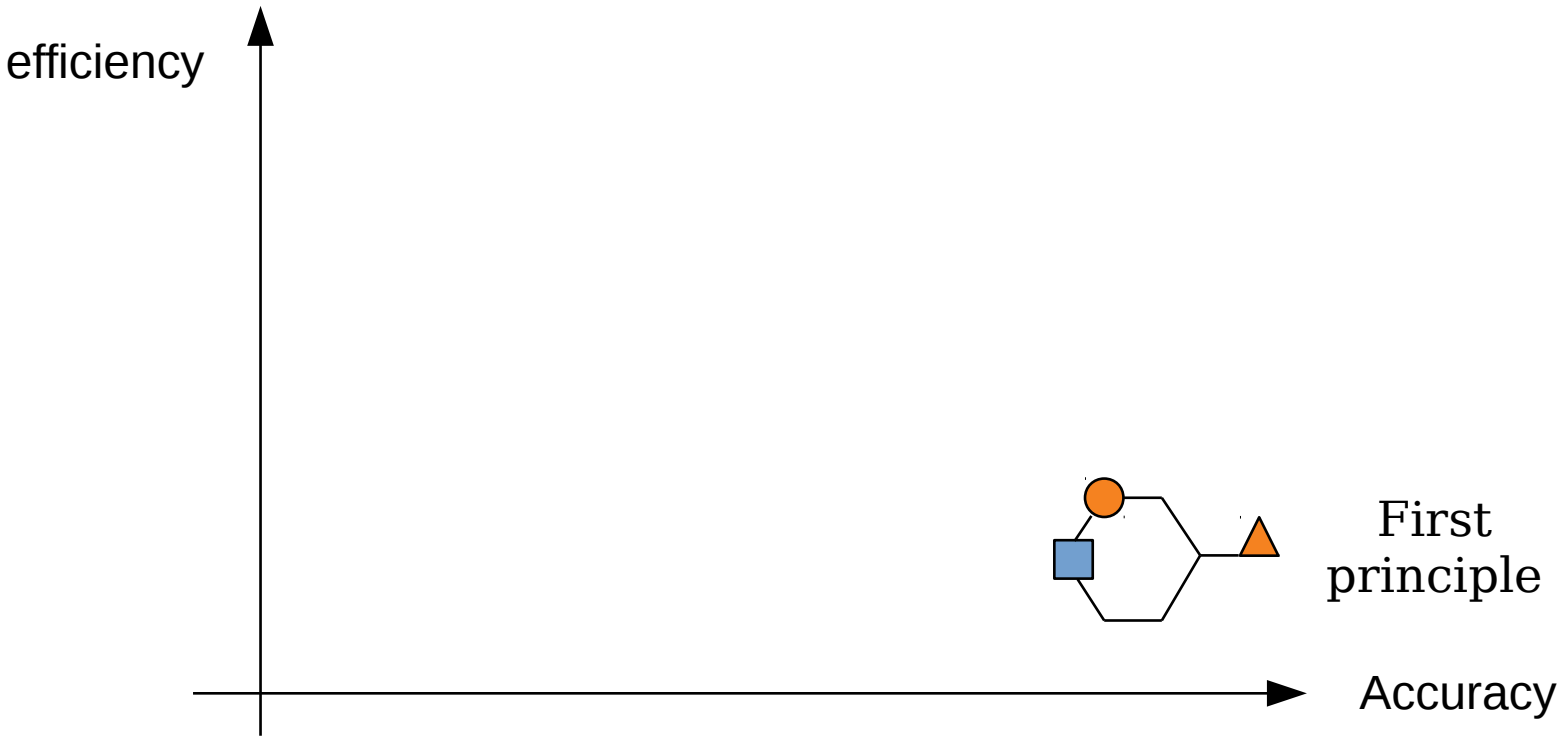
Coupled cluster single double (triple) CCSD(T)	$O(N^7)$
Second order Moller-Plesset (MP2)	$O(N^5)$
Hartree-Fock (HF)	$O(N^4)$
Density functional theory (DFT)	$O(N^3)$



Computationally expensive: DFT ~100 atoms ~100 picoseconds

Background: accuracy v.s. efficiency

Background: accuracy v.s. efficiency



Empirical force fields

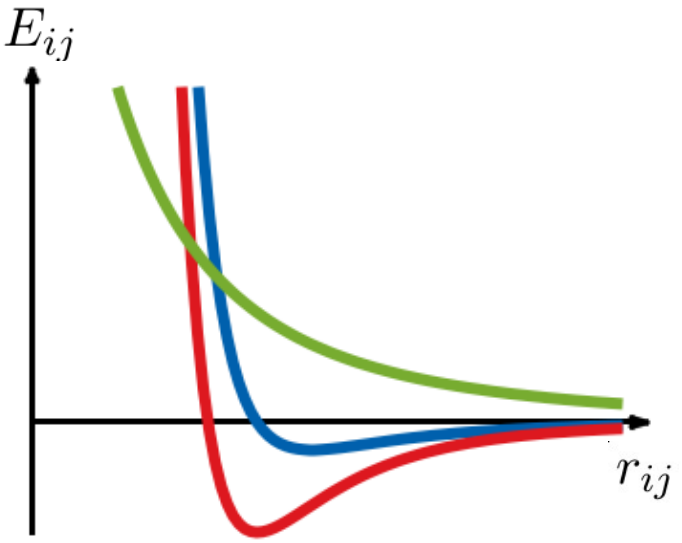
人为猜测势函数的函数形式，拟合其中的参数

$$E(r_1, r_2, \ldots, r_N)$$

难点：输入向量维度： $3N$

在特定的假设下，简化、分解高维问题为低维问题
例如 Charmm, Amber, Gromos, OPLS-AA 等

Background: empirical force field

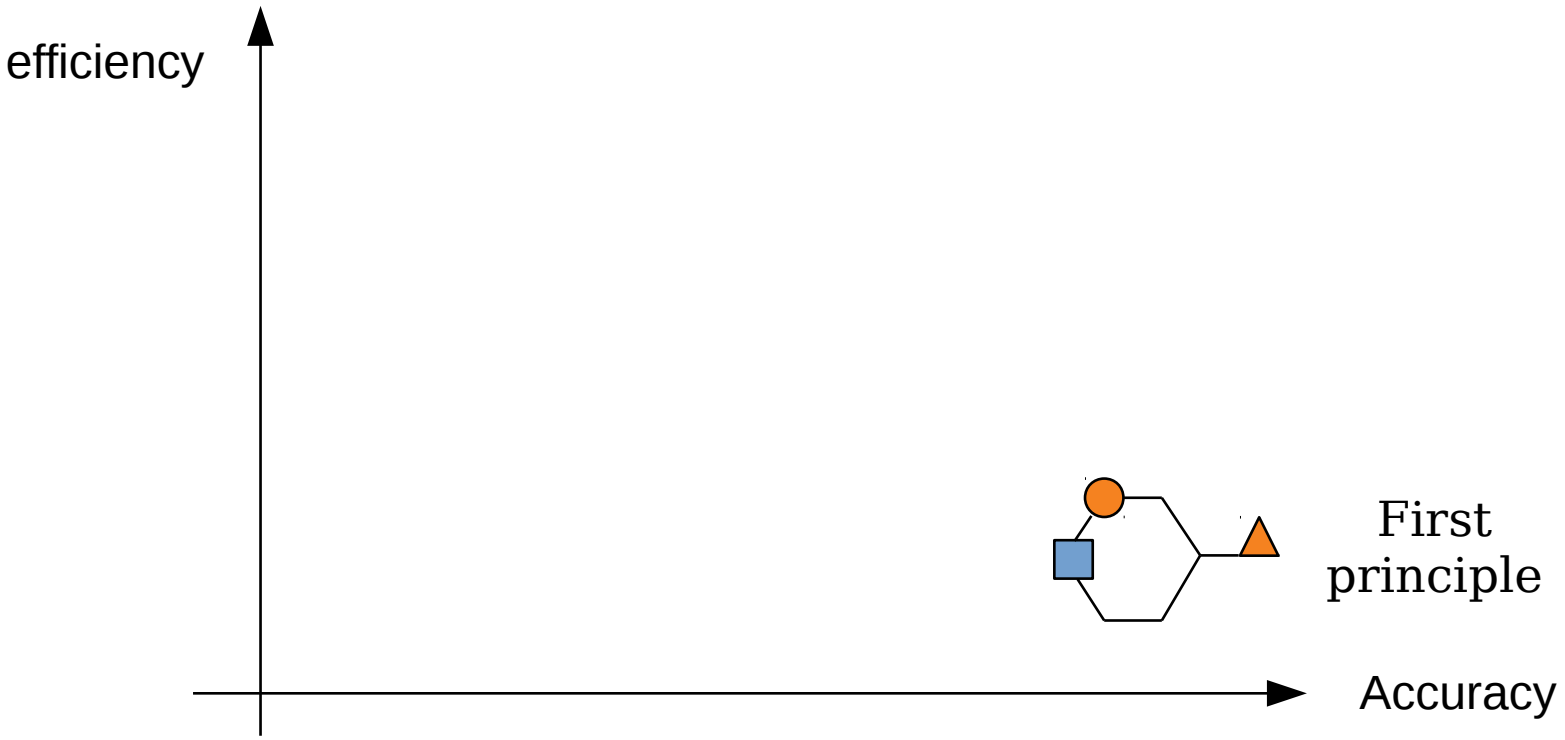


For example the Lennard-Jones potential for noble gas.

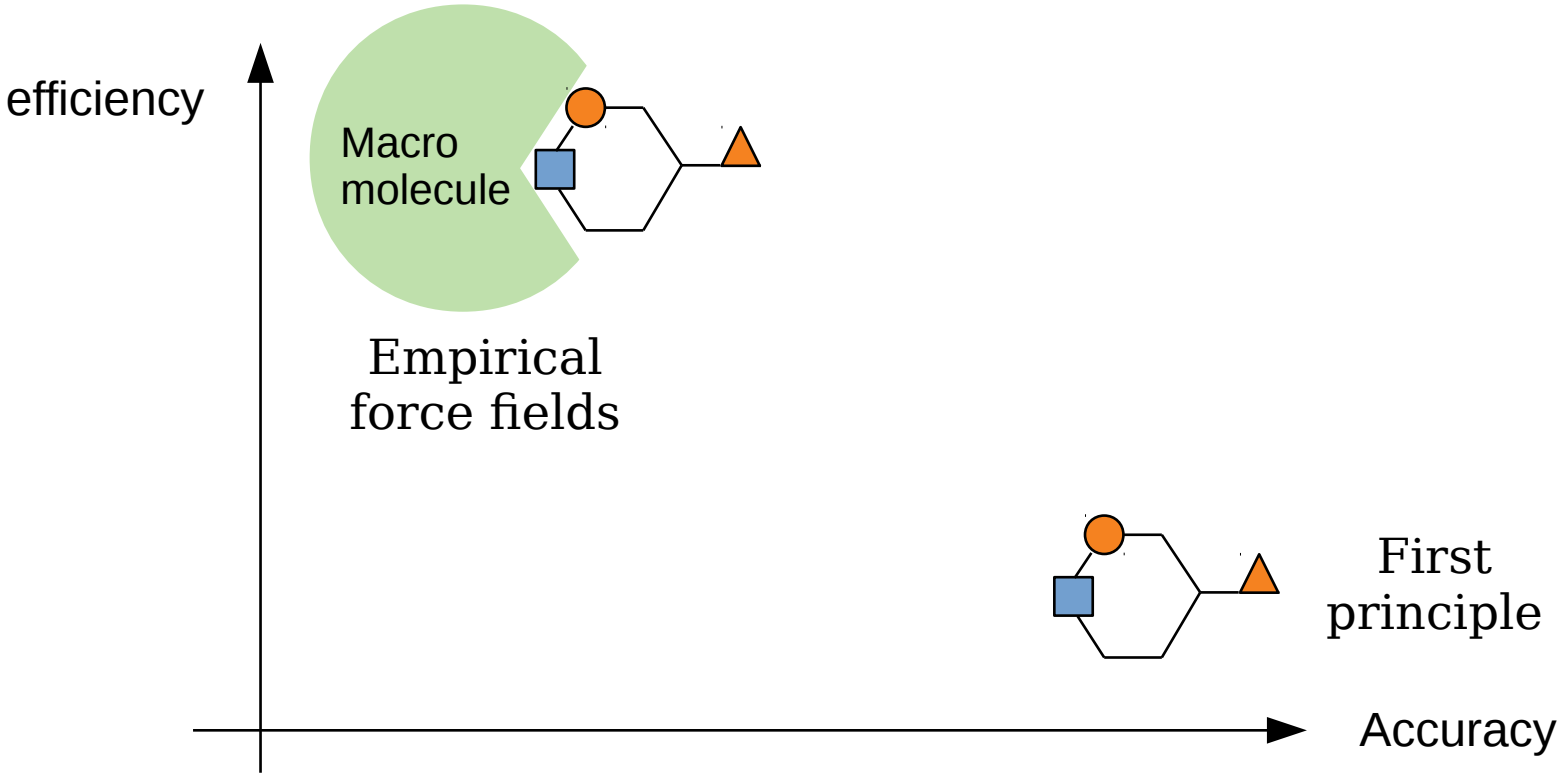
$$E = \frac{1}{2}\sum_{i \neq j} E_{ij}, \quad E_{ij} = 4\epsilon[(\frac{\sigma}{r_{ij}})^{12} - (\frac{\sigma}{r_{ij}})^{6}].$$

with ϵ and σ being tunable parameters

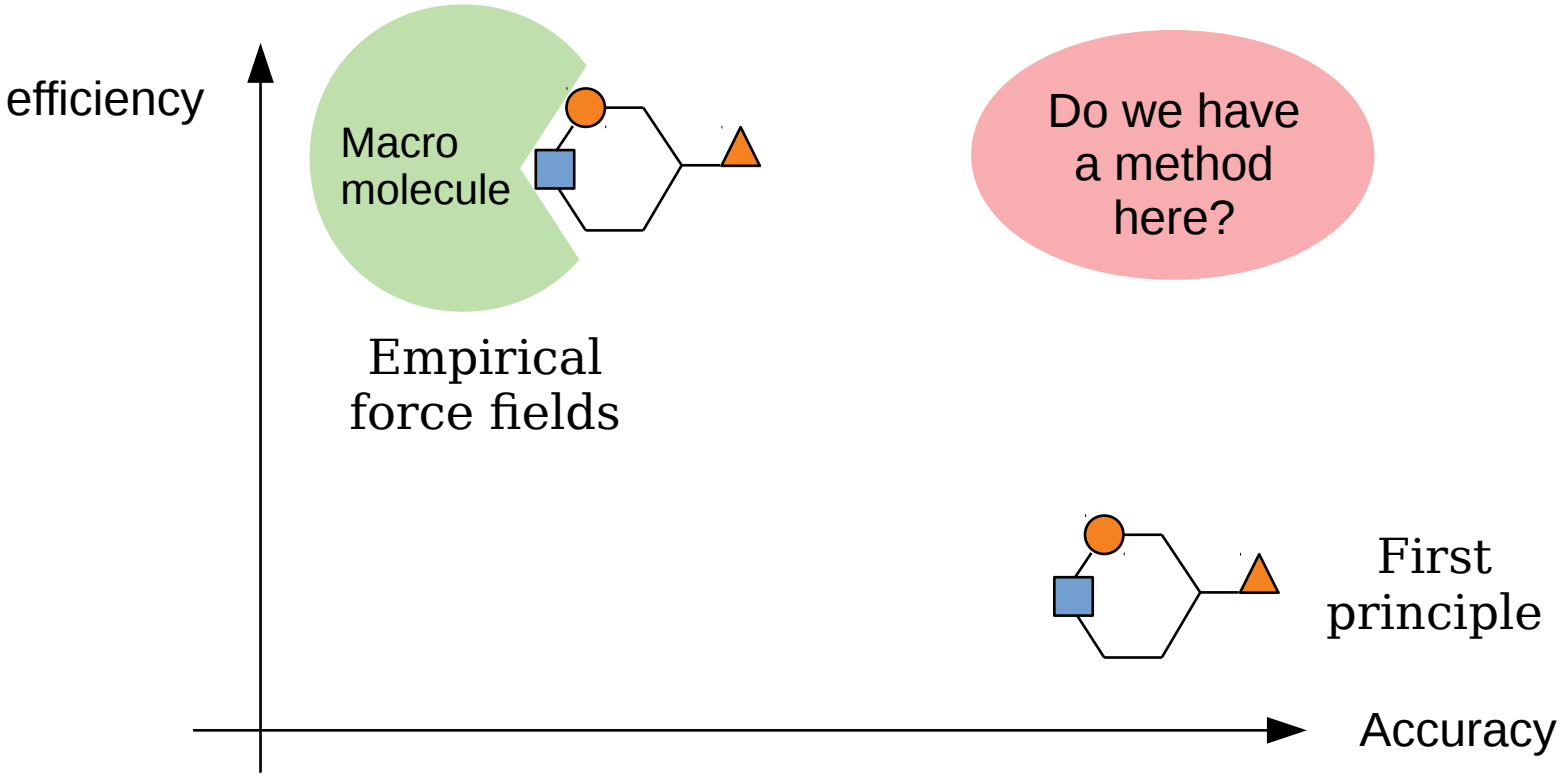
Background: accuracy v.s. efficiency



Background: accuracy v.s. efficiency



Background: accuracy v.s. efficiency



Background: difficulty in model potential

$$E = E(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N)$$

3N dimensional input

Background: difficulty in model potential

$$E = E(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N)$$

3N dimensional input

First principle calculations are super expensive.
Let the machine learn from the result, and act as a FP calculator?

Outline

A brief introduction do deep learning

Modeling the potential by deep learning

Data generation

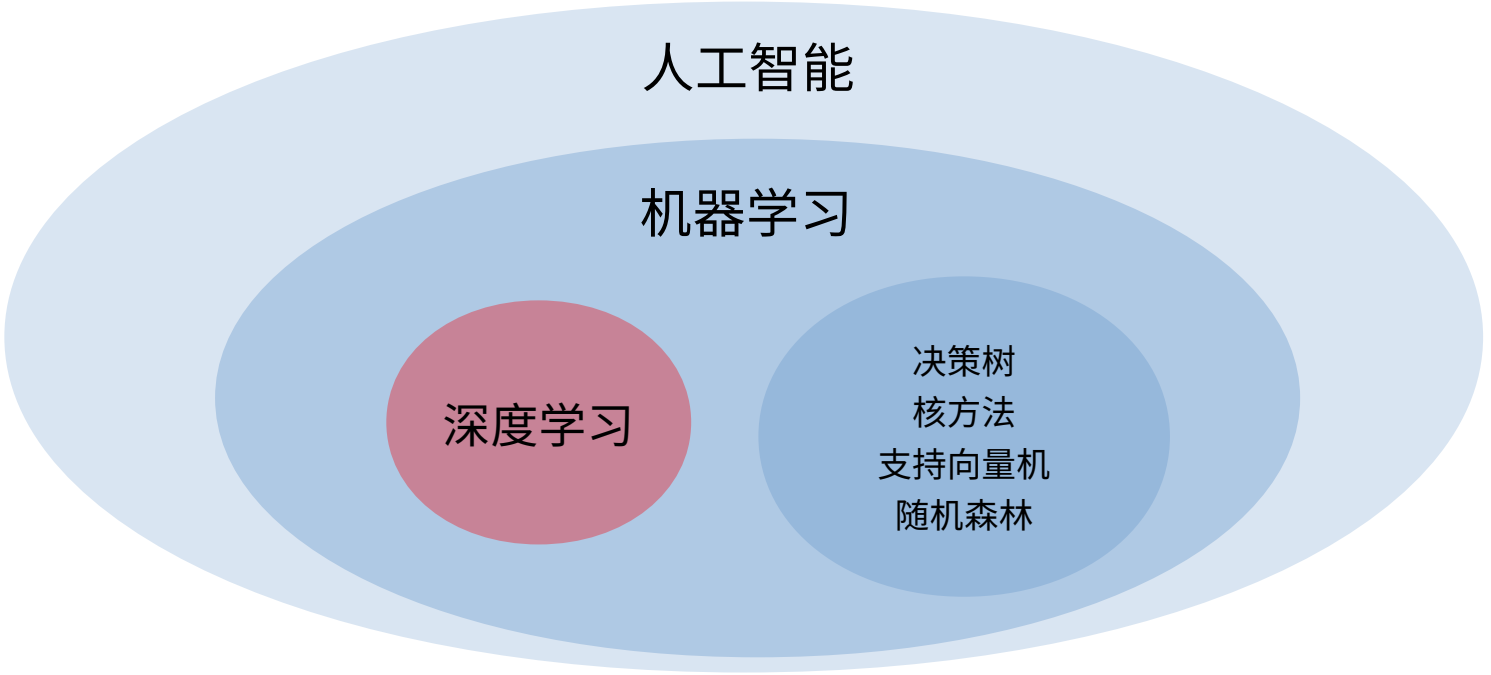
Outline

A brief introduction do deep learning

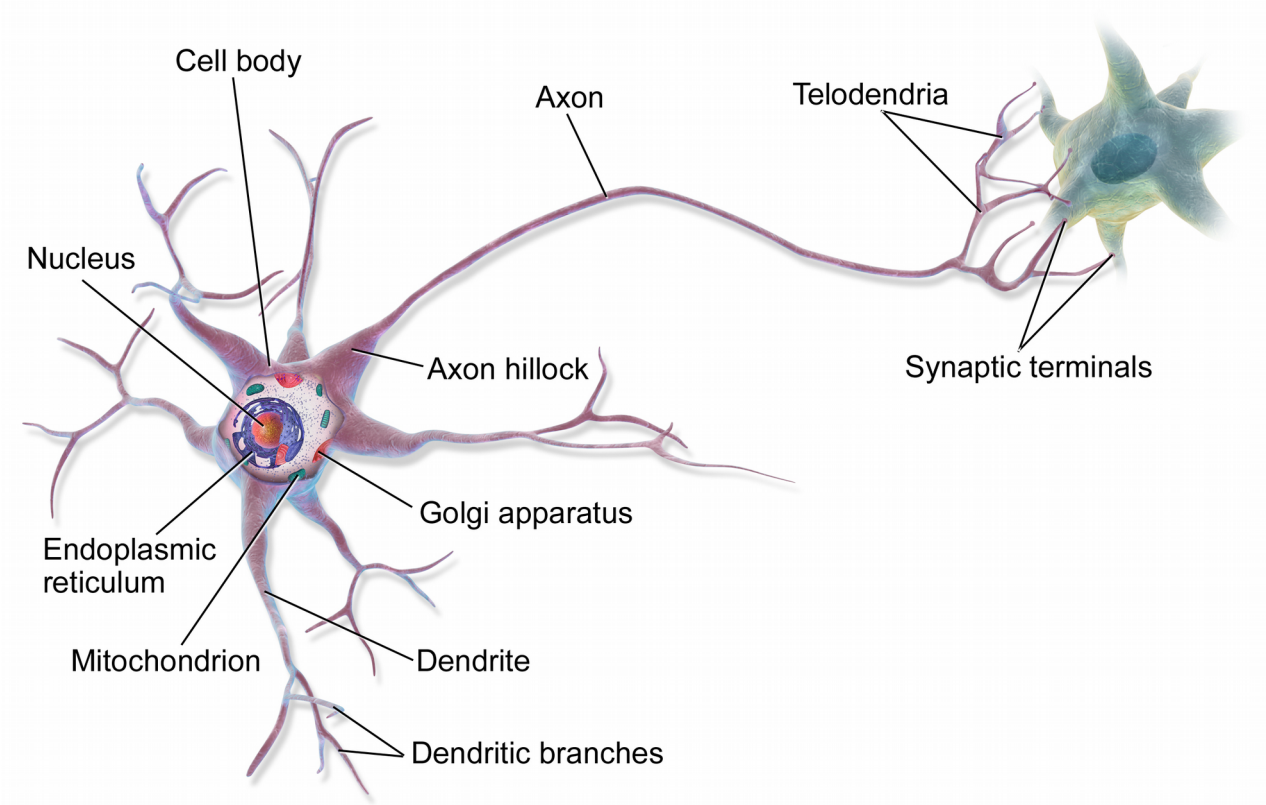
Modeling the potential by deep learning

Data generation

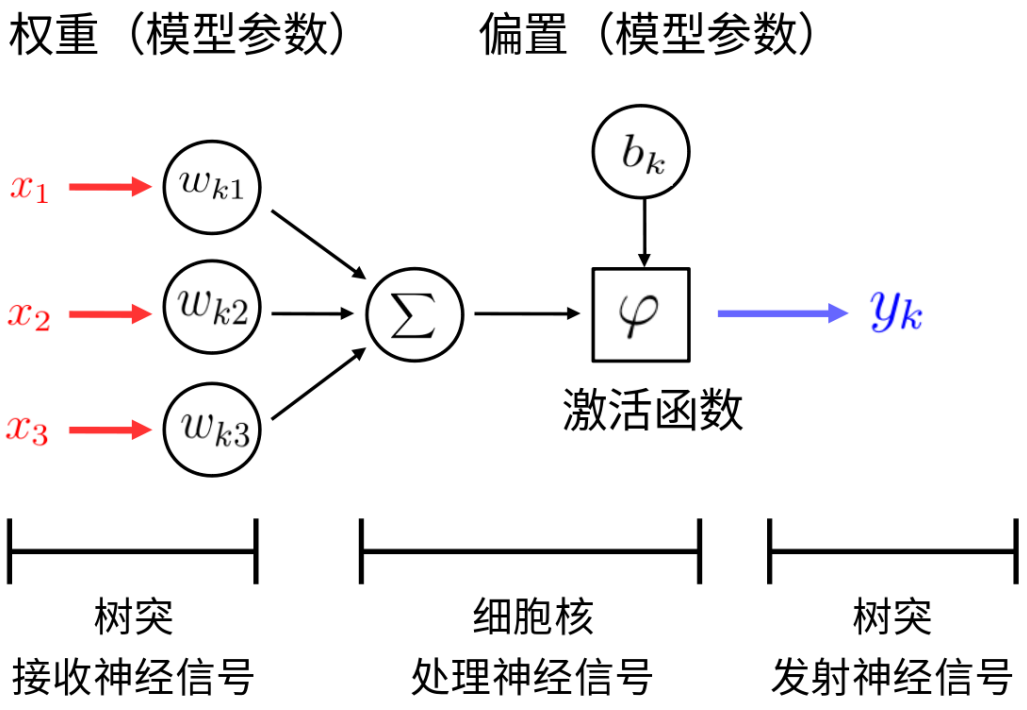
Artificial intelligence and machine learning

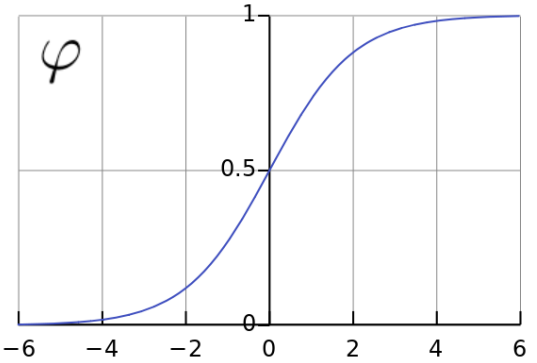


Deep learning: biological back ground



Deep learning: mathematical model



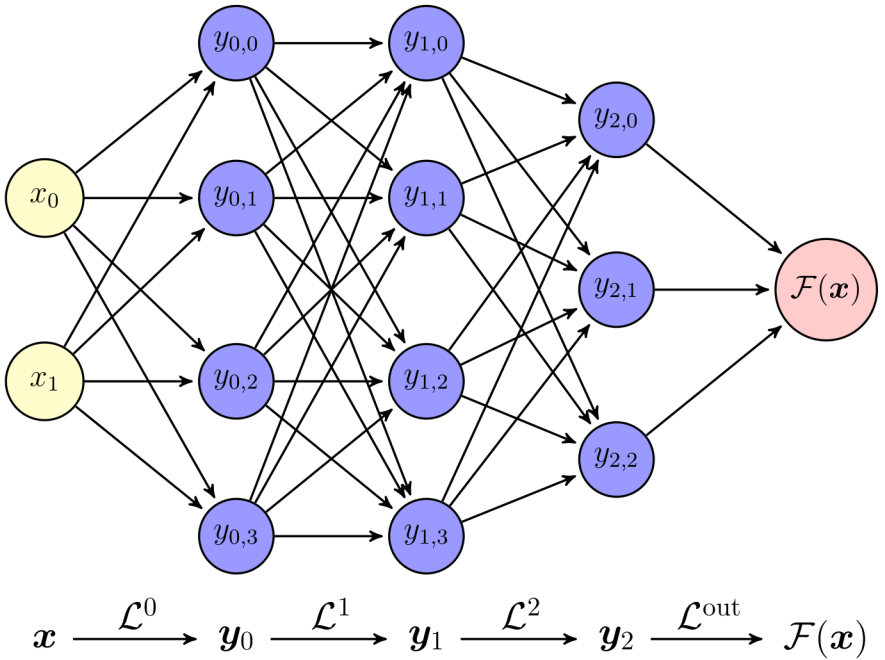


$$y_k = \varphi\left(\sum_i w_{ki}\, x_i + b_k\right)$$

上一层第 i 个神经元输入 x_i

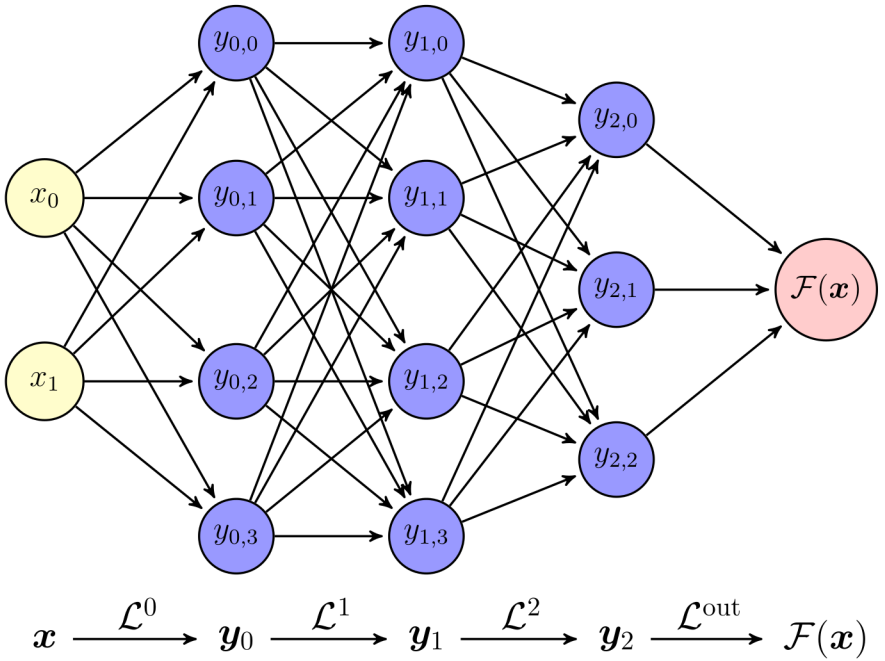
本层第 k 个神经元输出 y_k

Deep learning



$$\mathcal{F}(x) = W_3 \cdot \varphi(W_2 \cdot \varphi(W_1 \cdot \varphi(W_0 \cdot x + b_0) + b_1) + b_2) + b_3$$

Deep learning



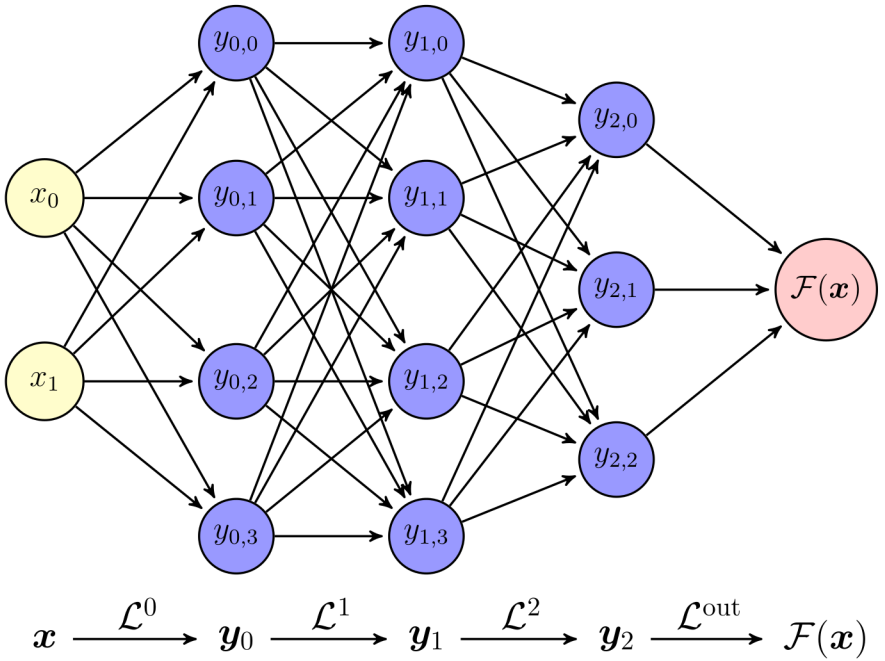
Deep learning:

$$\min_{\omega} \parallel \mathcal{F}(x, \omega) - f(x) \parallel$$

$$\omega = \{W_k, b_k\}$$

$$\mathcal{F}(x) = W_3 \cdot \varphi(W_2 \cdot \varphi(W_1 \cdot \varphi(W_0 \cdot x + b_0) + b_1) + b_2) + b_3$$

Deep learning



Flexibility to approximate functions

Especially for high-dimensional cases

Grid based methods: $O(N^d)$

$$\mathcal{F}(x) = W_3 \cdot \varphi(W_2 \cdot \varphi(W_1 \cdot \varphi(W_0 \cdot x + b_0) + b_1) + b_2) + b_3$$

Deep learning: universal approx. results

Shallow neural networks:

Barron, IEEE Transactions on Information theory, 39, 930 (1993)

Deep neural networks: (why deep is preferred over shallow)

Liang and Srikant, arXiv:1610.04161 (2016)

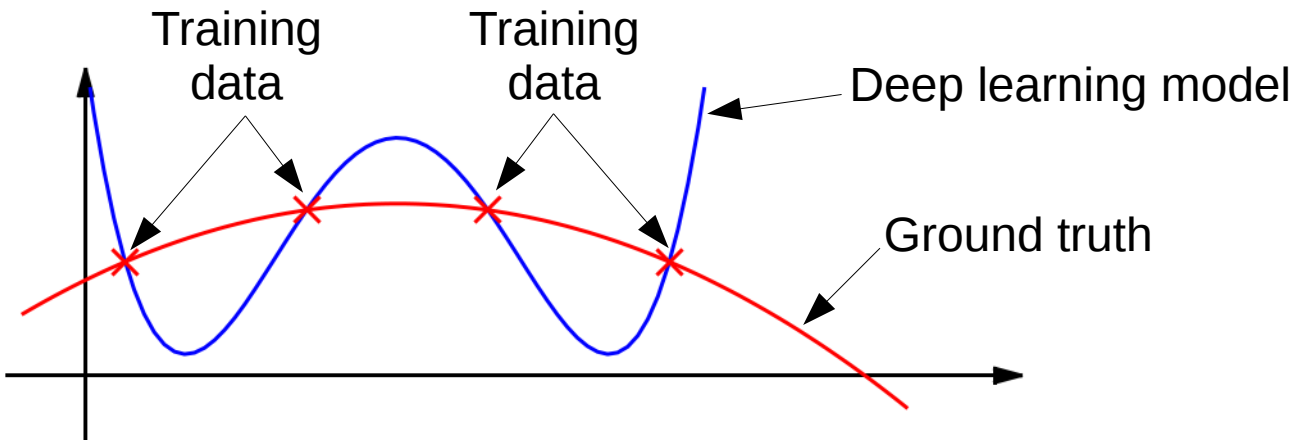
Yarotsky, Neural Networks, 94, 103 (2017)

Lu, Shen, Yang and Zhang, arXiv:2001.03040 (2020)

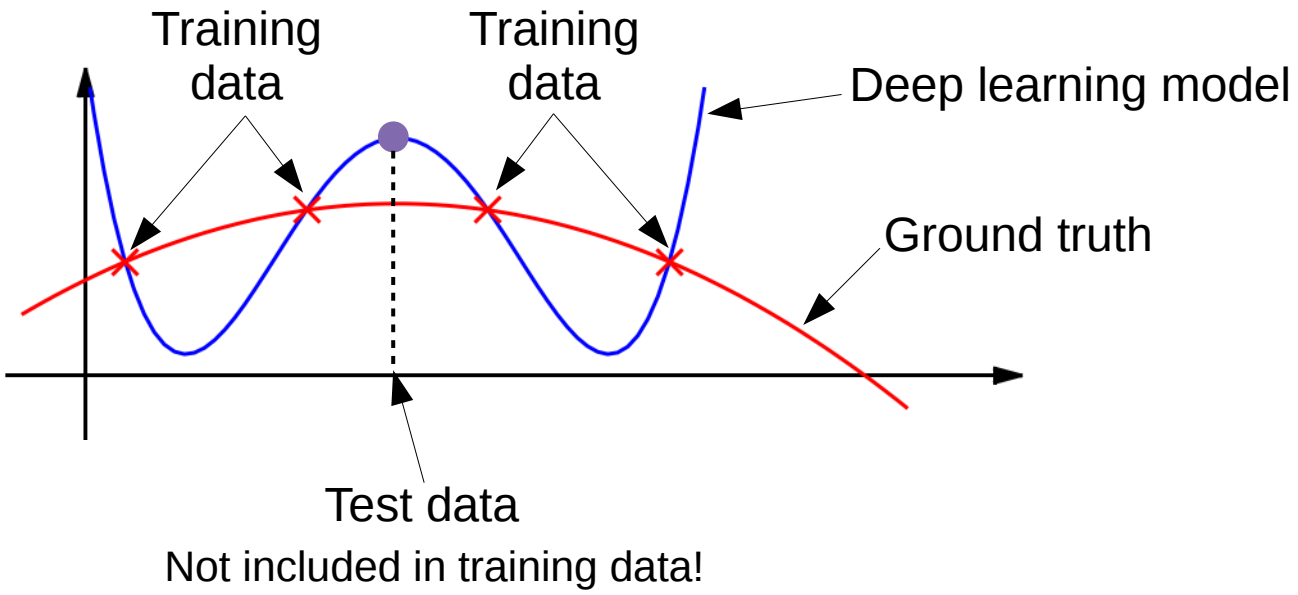
Curse of dimensionality:

E, Ma and Wu, arXiv:1810.06397 (2018) and arXiv 1906.08039 (2019)

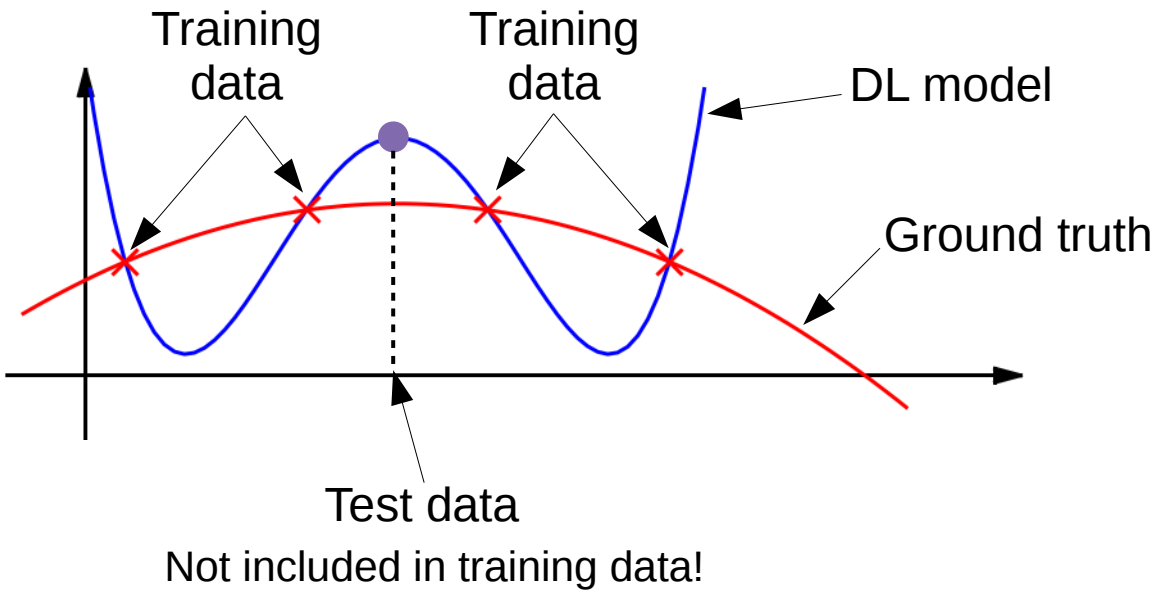
Deep learning: quality of the model



Deep learning: quality of the model



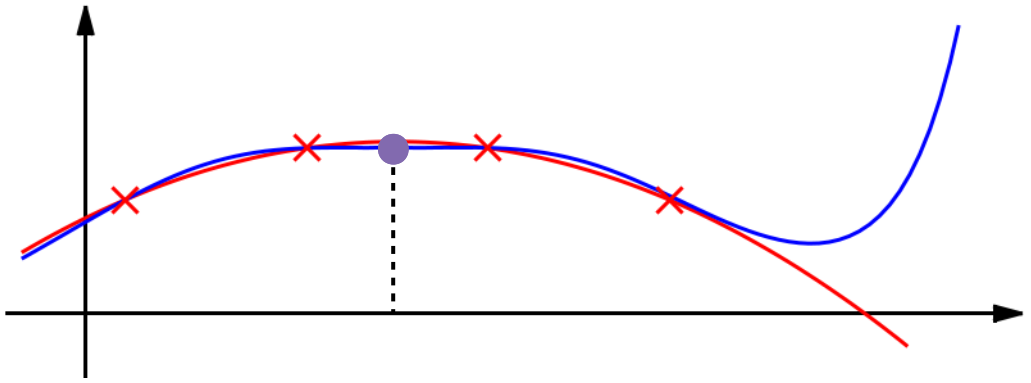
Deep learning: quality of the model



Over fitting model:

Test error much larger than training error

Deep learning: quality of the model



Properly fitting model:

Test error

≈

training error

Deep learning: quality of the model

	Training error	Test error
Under fitting	×	×
Over fitting	√	×
Properly fitting	√	√

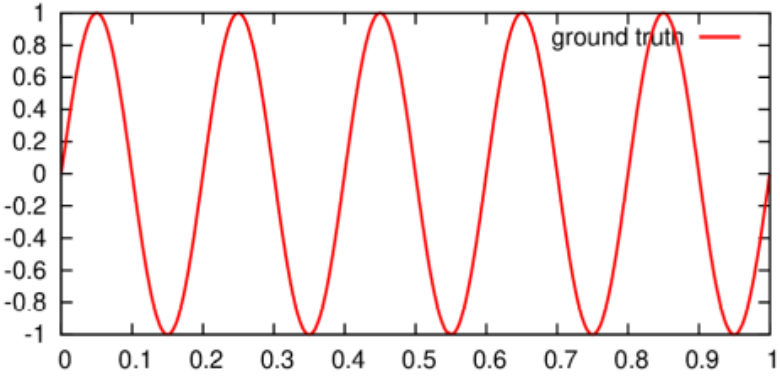
Deep learning: an example

学习目标 (ground truth) $f(x) = \sin(10\pi x)$ [0, 1] 区间

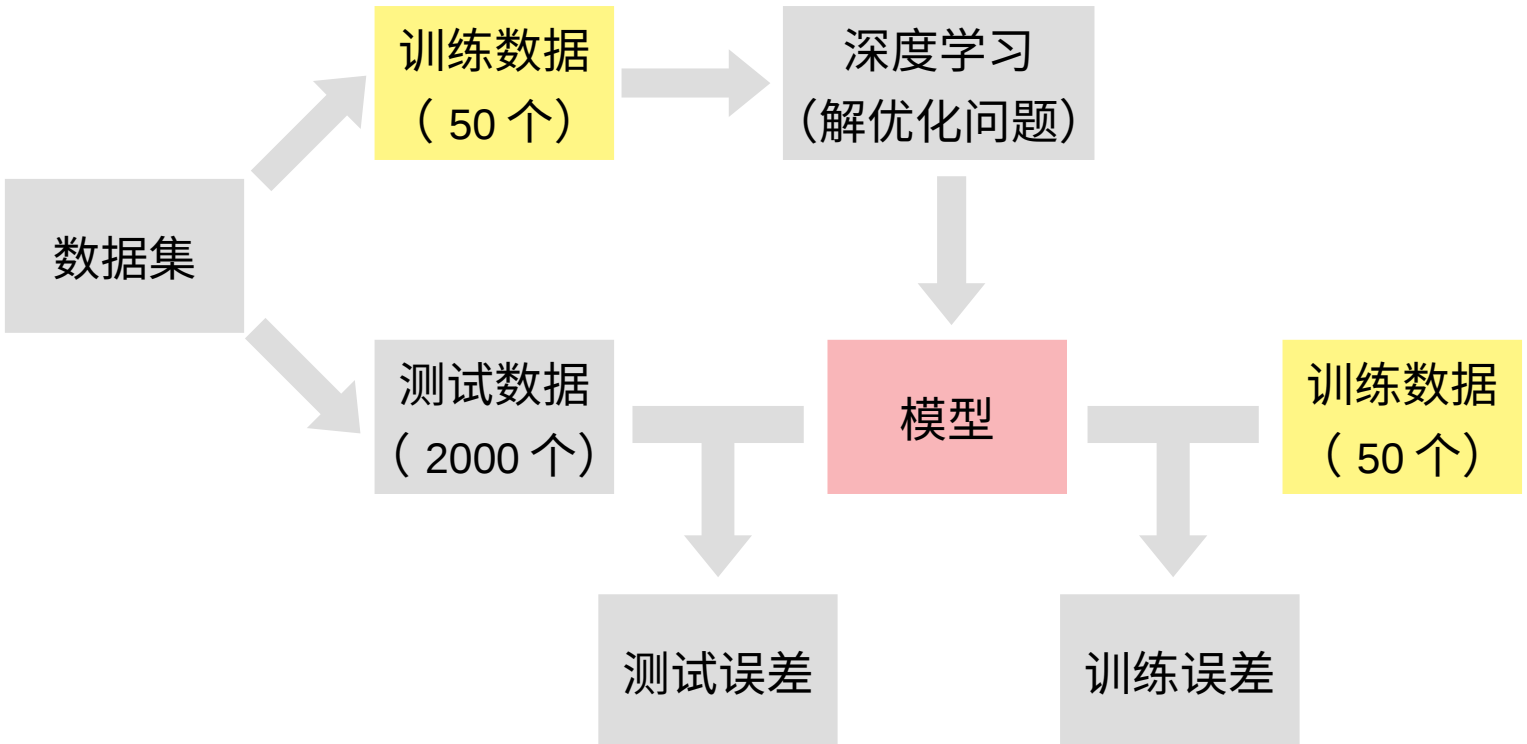
数据：$\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ x_i 在 [0,1] 均匀随机取样 $y_i = f(x_i)$

模型：$\mathcal{F}(x, w)$ 为深度神经网络 50 x 50 x 50 x 50

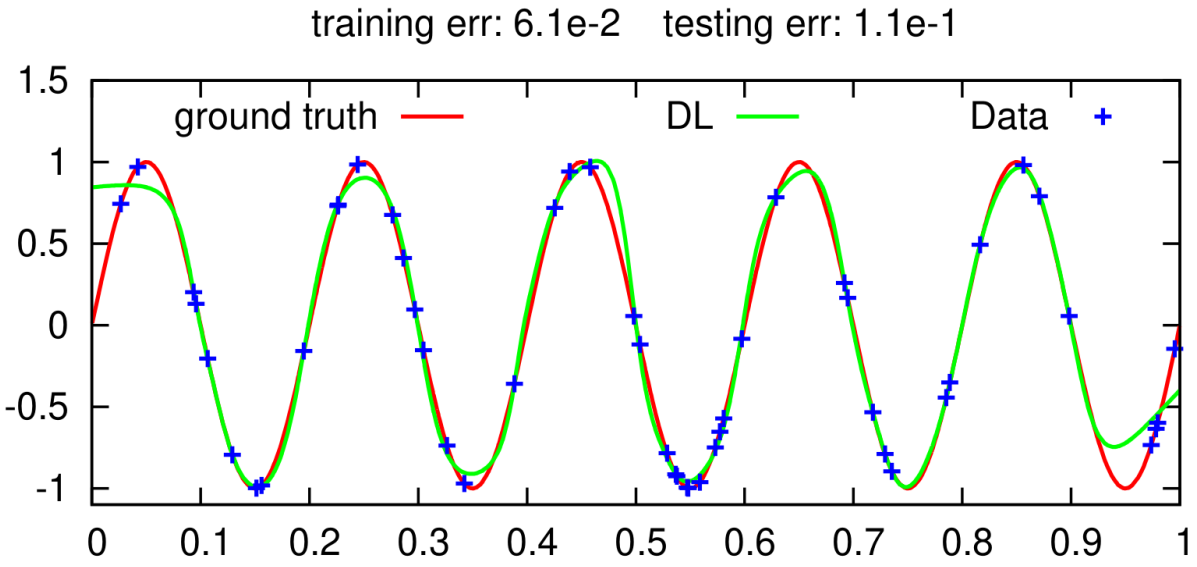
深度学习： $\min_w D(\mathcal{F}(x_i, w), y_i)$



Deep learning: an example

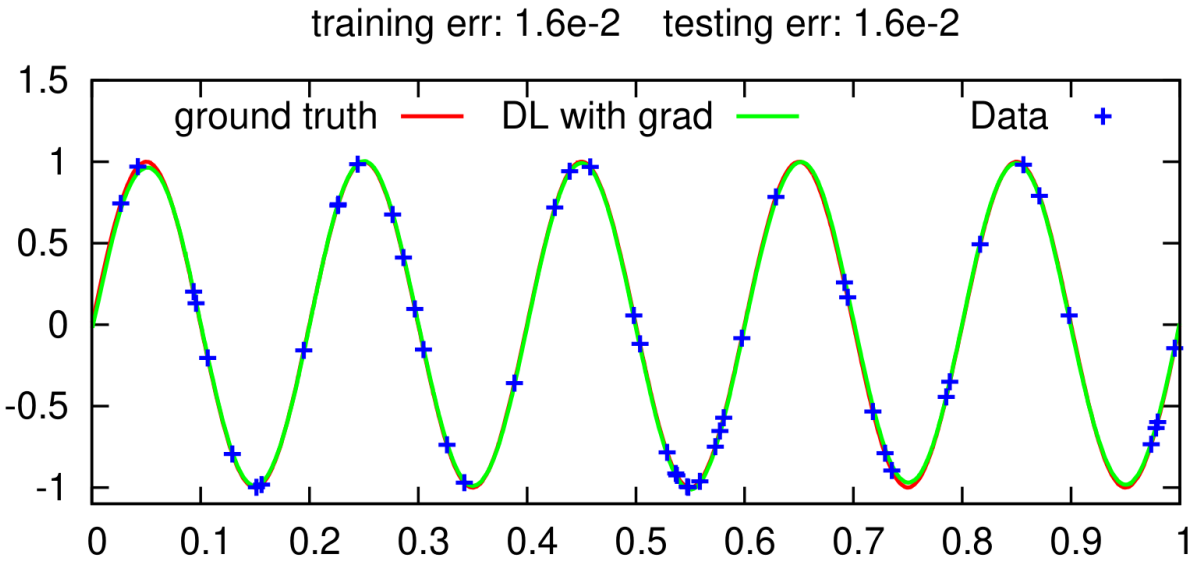


Deep learning: an example



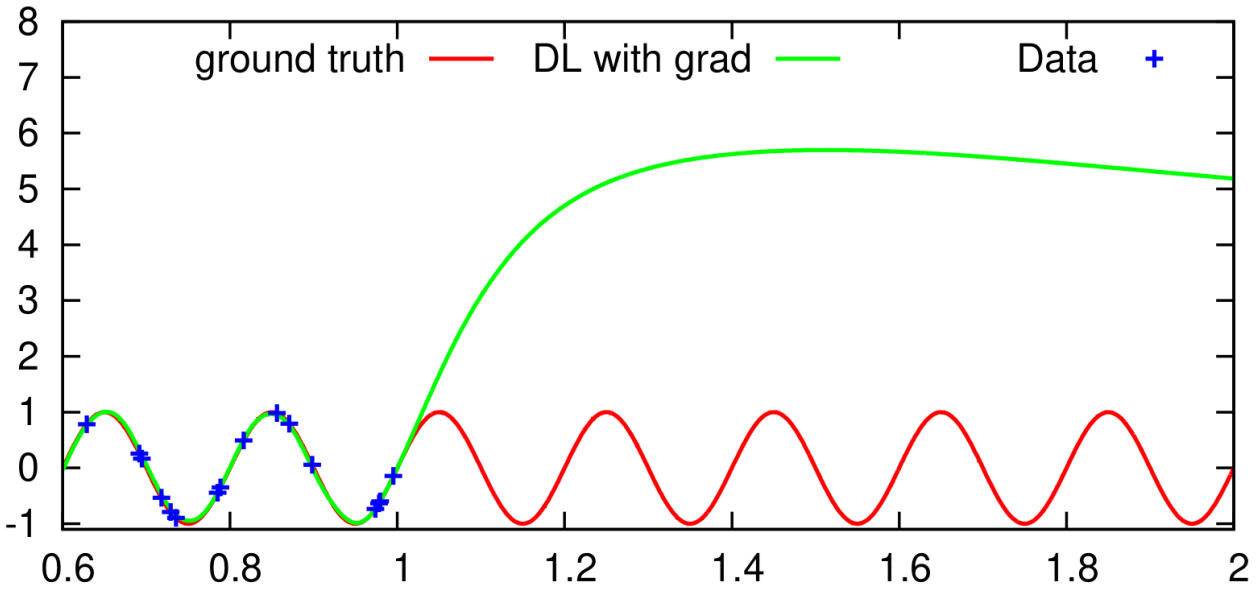
深度学习：$\| \mathcal{F}(x, w) - f(x) \|^2$

Deep learning: an example



深度学习：$\| \mathcal{F}(x,w) - f(x) \|^2 + \lambda \| \partial_x F(x,w) - \partial_x f(x) \|^2$

Deep learning: an example



NOT able to “extrapolate” at all

Outline

A brief introduction do deep learning

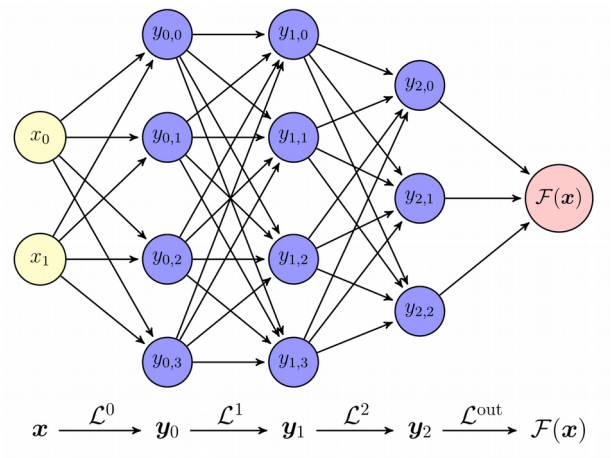
Modeling the potential by deep learning

Data generation

Learning an interatomic potential: setup

$$i\hbar\frac{\partial}{\partial t}|\Psi\rangle = H|\Psi\rangle$$

Learn the result of first principle cals.



Target: Quantum mechanical energy

$E(\boldsymbol{R})$, $\boldsymbol{R} = \{\boldsymbol{r}_i\}$ dimension: 3N

Model: $E(\boldsymbol{R}, \omega)$

Data: $\{(\boldsymbol{R}_0, E_0), (\boldsymbol{R}_1, E_1), (\boldsymbol{R}_2, E_2), \dots\}$

Given $\boldsymbol{R}_i$, $E_i = E(\boldsymbol{R}_i)$ computed by quantum mechanical method

Training: $\min_{\omega} \parallel E(\boldsymbol{R}_i, \omega) - E_i \parallel^2$

Extensibility

Trained potential:

$$E = E(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N) \approx \mathrm{DNN}(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N)$$

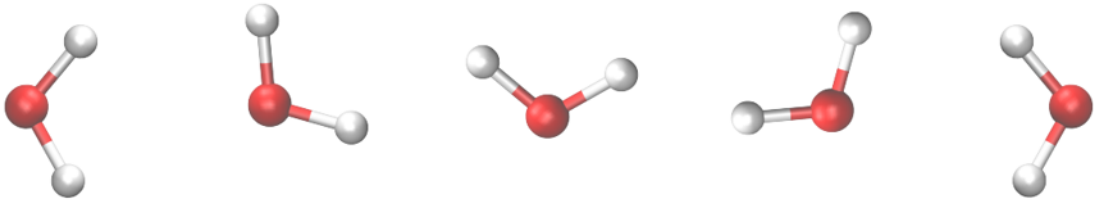
DNN == Deep Neural Network

One wants to evaluate:

$$E(\boldsymbol{r}_1, \ldots, \boldsymbol{r}_i, \ldots, \boldsymbol{r}_N, \boldsymbol{r}_{N+1})$$

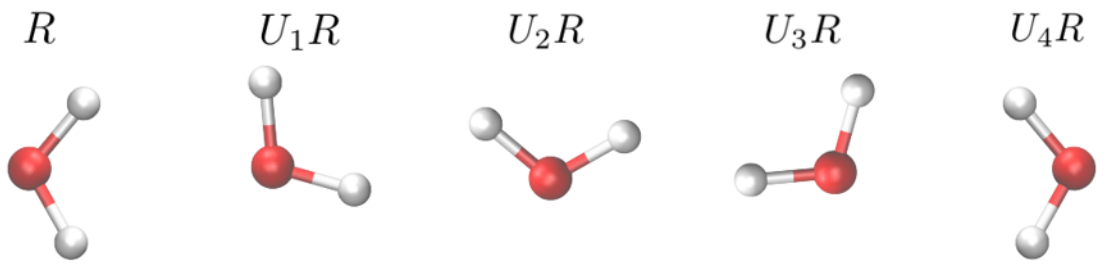
Rotational symmetry

R U_1R U_2R U_3R U_4R



same E

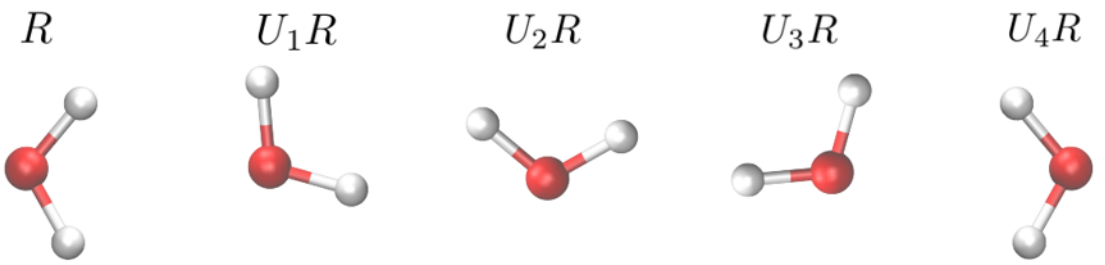
Rotational symmetry



same E

$$\{(R, E), (U_1R, E), (U_2R, E), \dots\}$$

Rotational symmetry

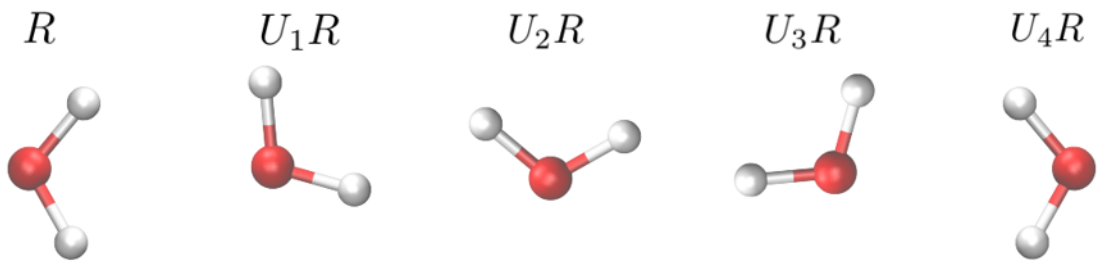


same E

$$E(R) = E(UR)$$

$$\{(R, E), (U_1R, E), (U_2R, E), \dots\}$$

Rotational symmetry



same E

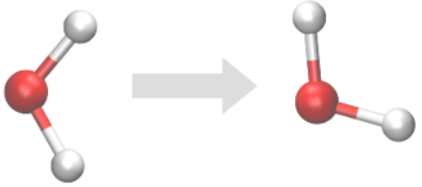
$$\{(R,E),(U_1R,E),(U_2R,E),\dots\} \xrightarrow{E(R)=E(UR)} \{(R,E)\}$$

Symmetries in a molecular system

Translational

$$E(r_0, r_1, r_2) = E(r_0+s, r_1+s, r_2+s)$$

Rotational

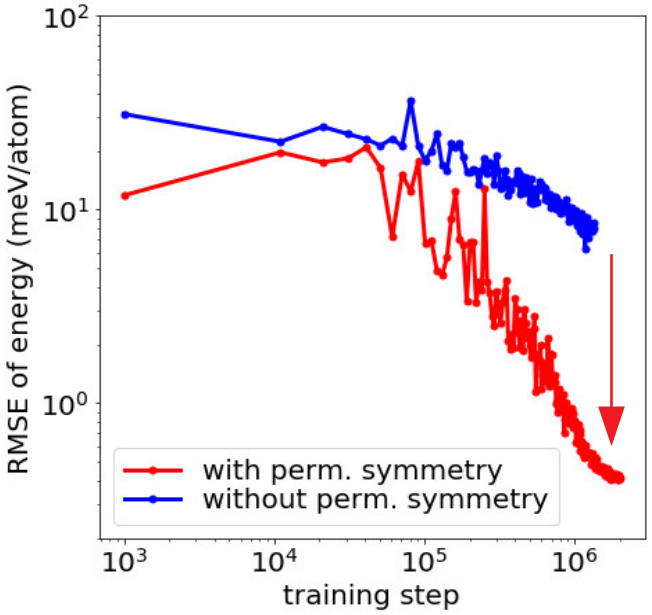


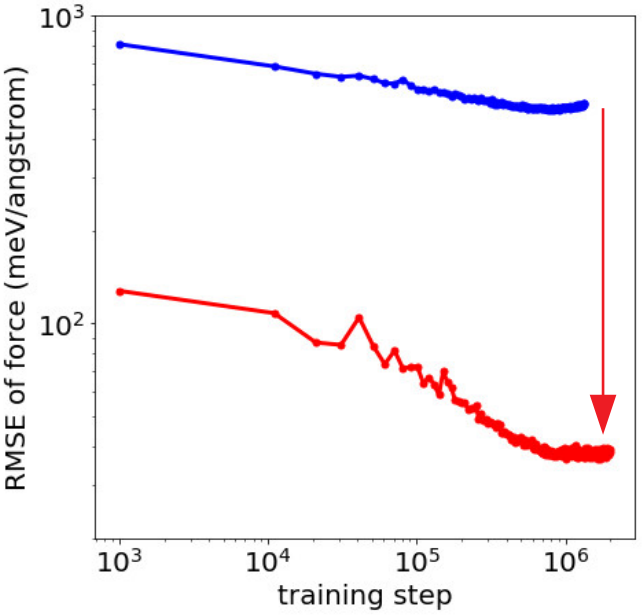
$$E(r_0, r_1, r_2) = E(c + U(r_0 - c), c + U(r_1 - c), \cdots)$$

permutational

$$E(r_0, r_1, r_2) = E(r_0, r_2, r_1)$$

The advantage of symmetry preserving





A good machine learning model should be

Extensible

Respect the physical symmetries

Accuracy and efficiency

Minimal human intervention

Construction of machine learning potentials

$$E = \sum_i E_i$$

Construction of machine learning potentials

$$E = \sum_i E_i$$

$$E_i = \mathcal{F}(\mathcal{D}(R_i))$$

$\mathcal{F}$ Fitting model

$\mathcal{D}$ Descriptor: preserves symmetries

$$\mathcal{D}(R) = \mathcal{D}(UR)$$

$$R_i = \{r_i\} \bigcup \{r_j : |r_j - r_i| \leq r_c\}$$

A summary of machine learning models

Model	Descriptor	Fitting	Data scal.	System scal.	Human interven.	Reference
SOAP	SOAP	Kernel	no	yes	some	PRB 2013
GDML	Coulomb matr.	Kernel	no	no	some	Sci. Adv. 2017
DTNN	Coulomb matr.	NN	yes	no	some	Nat.Comm 2017
BPNN	Symm. funcs.	NN	yes	yes	heavy	PRL 2007
SchNet	Schnet	NN	yes	yes	no	NeurIPS 2017
DeepPot	Deep Pot.	NN	yes	yes	no	NeurIPS 2018

Deep potential: design of descriptor

(Zaheer 2017 Deep Sets)

Function $f(x_1, x_2, \ldots x_N)$ is permutationally invariant if and only if there exist ρ and ϕ such that $f(\{x_i\}) = \rho(\sum_i \phi(x_i))$

Deep potential: design of descriptor

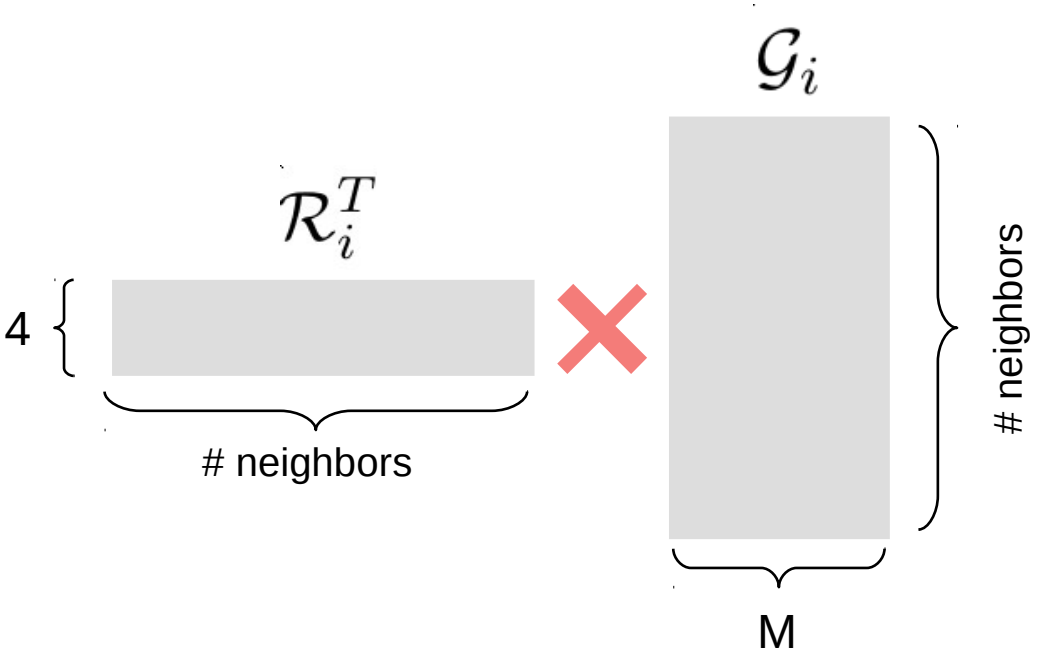
Learned by DNN

$$\mathcal{R}_i = \begin{pmatrix} 1/r_{i1} & x_{i1}/r_{i1}^2 & y_{i1}/r_{i1}^2 & z_{i1}/r_{i1}^2 \\ 1/r_{i2} & x_{i2}/r_{i2}^2 & y_{i2}/r_{i2}^2 & z_{i2}/r_{i2}^2 \\ 1/r_{i3} & x_{i3}/r_{i3}^2 & y_{i3}/r_{i3}^2 & z_{i3}/r_{i3}^2 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \qquad \mathcal{G}_i = \begin{pmatrix} G_1(r_{i1}) & G_2(r_{i1}) & G_3(r_{i1}) & \cdots \\ G_1(r_{i2}) & G_2(r_{i2}) & G_3(r_{i2}) & \cdots \\ G_1(r_{i3}) & G_2(r_{i3}) & G_3(r_{i3}) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

$\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j \quad r_{ij} = (x_{ij}, y_{ij}, z_{ij})$ Hooray! Translational invariant

$r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$ Translational and rotational invariant

Deep potential: design of descriptor



$$(\mathcal{R}_i^T \mathcal{G}_i)_{1k} = \sum_j G_k(r_{ij}) \frac{1}{r_{ij}^2}$$

$$(\mathcal{R}_i^T \mathcal{G}_i)_{2k} = \sum_j G_k(r_{ij}) \frac{x_{ij}}{r_{ij}^2}$$

$$(\mathcal{R}_i^T \mathcal{G}_i)_{3k} = \sum_j G_k(r_{ij}) \frac{y_{ij}}{r_{ij}^2}$$

$$(\mathcal{R}_i^T \mathcal{G}_i)_{4k} = \sum_j G_k(r_{ij}) \frac{z_{ij}}{r_{ij}^2}$$

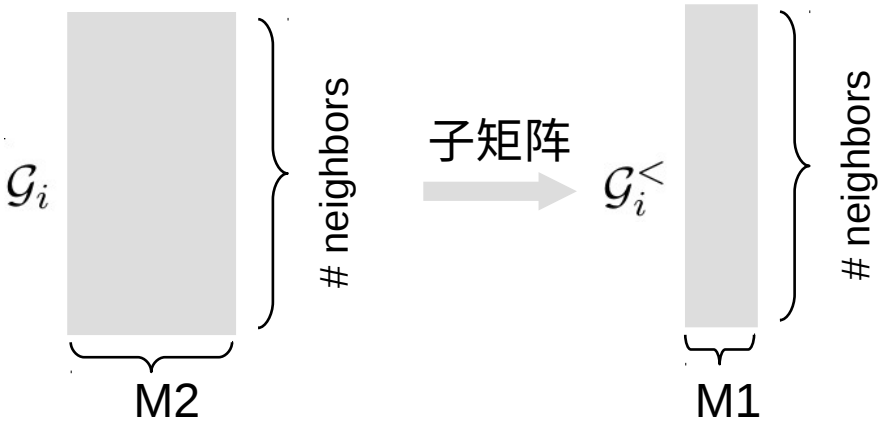
Remind: $f(\{x_j\}) = \rho\Big(\sum_j \phi(x_j)\Big)$

Deep potential: design of descriptor

Consider

$$\mathcal{D}_i = (\mathcal{G}_i^<)^T \mathcal{R}_i (\mathcal{R}_i)^T \mathcal{G}_i$$

which is permutationally invariant



Deep potential: design of descriptor

Consider

$$\mathcal{D}_i = (\mathcal{G}_i^<)^T \mathcal{R}_i (\mathcal{R}_i)^T \mathcal{G}_i$$

which is permutationally invariant

For any rotaional transform U, $G_k(r_{ij})$ is invariant under U, and

$$\tilde{\mathcal{D}}_i = (\mathcal{G}_i^<)^T (\mathcal{R}_i U)(\mathcal{R}_i U)^T \mathcal{G}_i = (\mathcal{G}_i^<)^T \mathcal{R}_i U U^T \mathcal{R}_i^T \mathcal{G}_i = \mathcal{D}_i$$

therefore, $\mathcal{D}_i$ is rotationally invariant

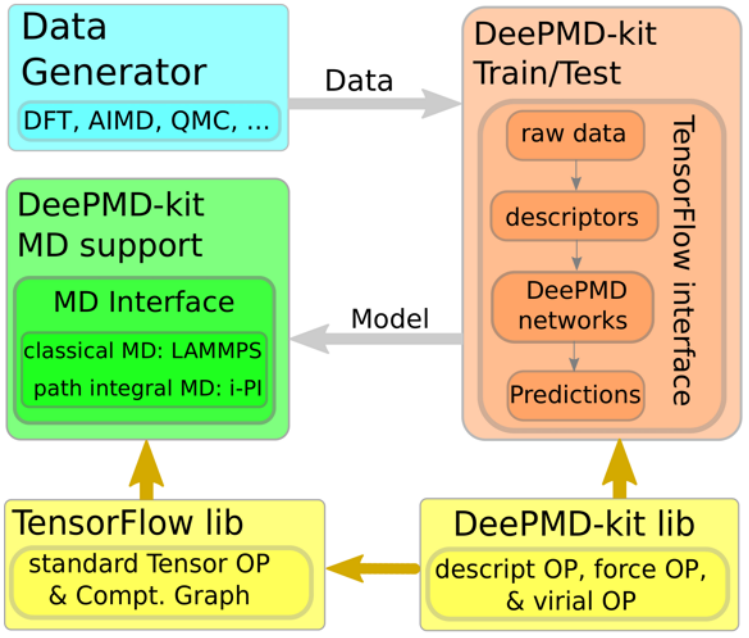
Deep potential

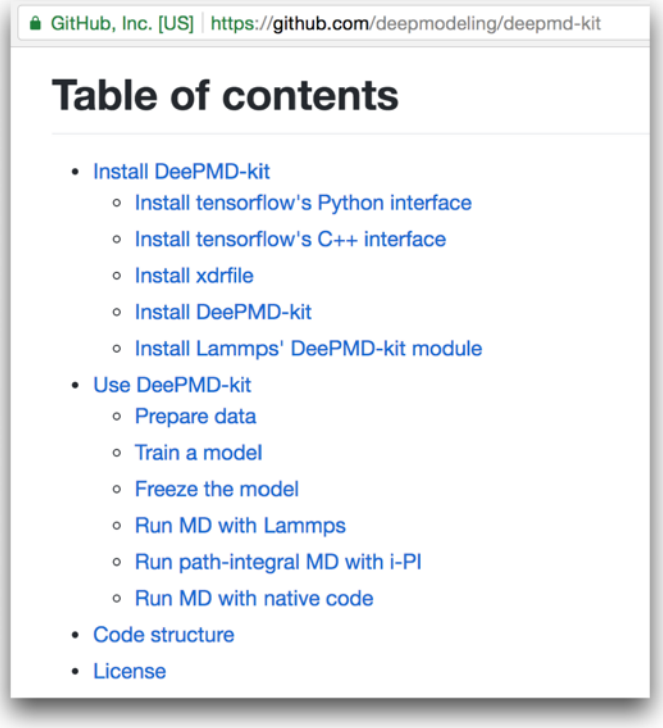
The deep potential: $E = \sum_i \mathcal{N}_{\alpha_i}\left(\mathcal{D}_{\alpha_i}(r_i, \{r_j\}_{j\in n(i)})\right)$

The force on each atom: $\boldsymbol{F}_i = -\nabla_{r_i} E$

The loss $\min_{\omega} \mathcal{L}$, $\mathcal{L} = p_e(E - \hat{E})^2 + p_f \sum_i |\boldsymbol{F}_i - \hat{\boldsymbol{F}}_i|^2.$

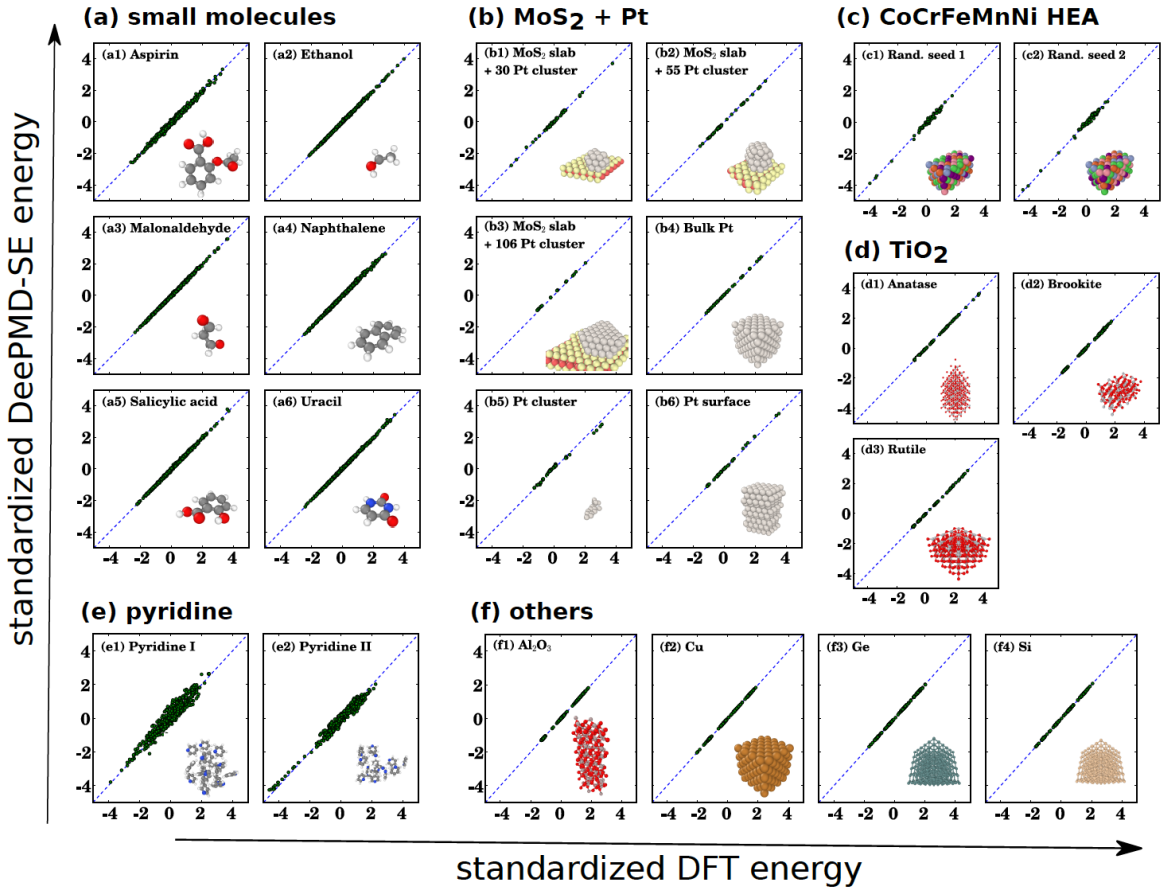
DeePMD-kit: an implementation of deep potential





Free software from https://github.com/deepmodeling/deepmd-kit

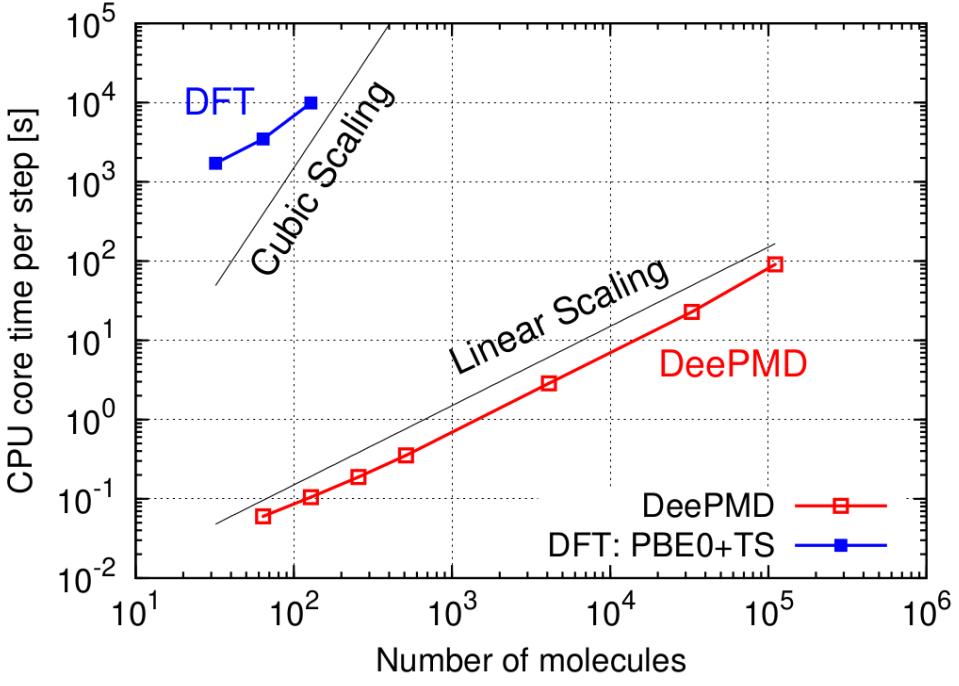
Accuracy comparable with first principle



DeePMD-kit: performance in training

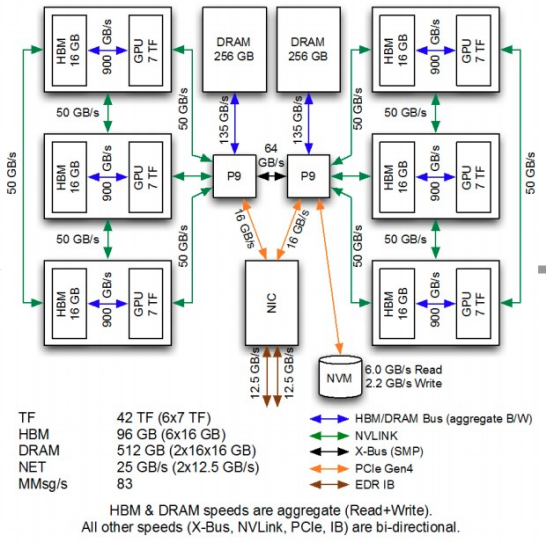
System	descriptor	#step	hardware	Time[hours]
water/ice	local frame	1×10^6	CPU Intel i7-6700HQ	~ 19
CG-water	local frame	1×10^6	CPU Intel i7-6700HQ	~ 18
Mg-Al	smooth	4×10^5	CPU Intel E5-2680v4	~ 11
Mg-Al	smooth	4×10^5	GPU Nvidia P100	~ 1.5

DeePMD-kit: performance in molecular dynamics

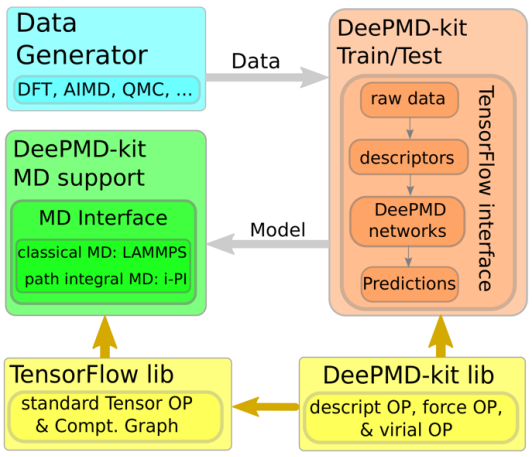


DeePMD-kit: optimization for super computers

DeePMD-kit
适应异构
架构的优化



并行可扩展
至 Summit 全机
（6.8 亿原子）



Outline

A brief introduction do deep learning

Modeling the potential by deep learning

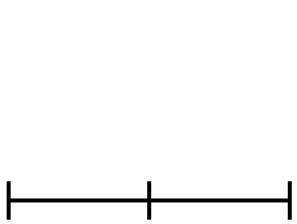
Data generation

Data: how?

$$\{(\boldsymbol{R}_0, E_0), (\boldsymbol{R}_1, E_1), (\boldsymbol{R}_2, E_2), \dots\}$$

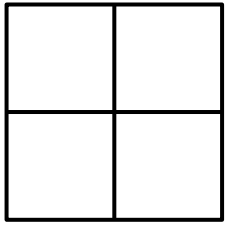
Which coordinate $\boldsymbol{R}_i$ to learn?

Data: uniform grid for approximating a function



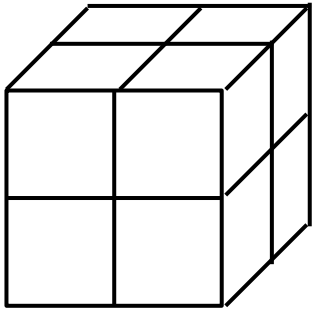
1 dimensional grid

N^1



2 dimensional grid

N^2



3 dimensional grid

N^3

.....

?

d dimensional grid

N^d

Data: uniform grid for approximating a function

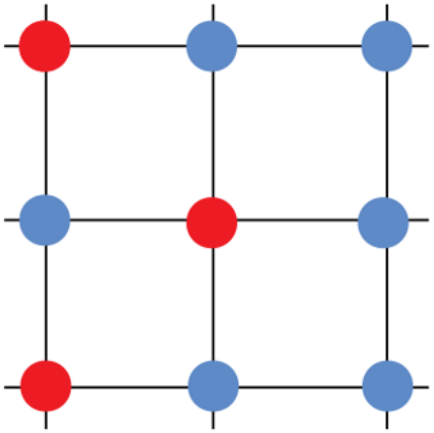
Target $E_i = \mathcal{N}_{\alpha_i}(\mathcal{D}_i)$. A typical dimension of $\mathcal{D}$ is 100.

A uniform grid, each dimension has N grid points, total number of grid point is N^d.

Curse of dimensionality

#Data (N^d)	10^4	10^6	10^8	10^{10}
#Dim (d)	100	100	100	100
#Grid points (N)	1.10	1.15	1.20	1.26

Data: complexity introduced by chemical species



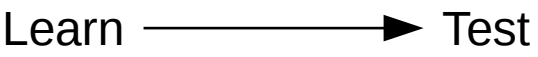
9 lattice points, each point can be occupied by either Al or Mg.
In total $2^9 = 512$ possibilities.

In gneral M^N possibilities, M number of species, N number of lattice points.

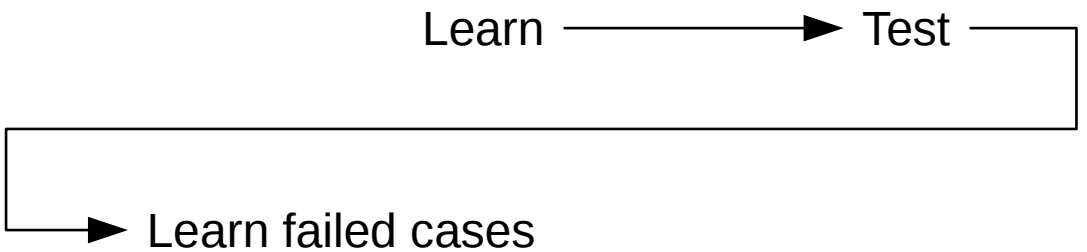
Data: how human learns?

Learn

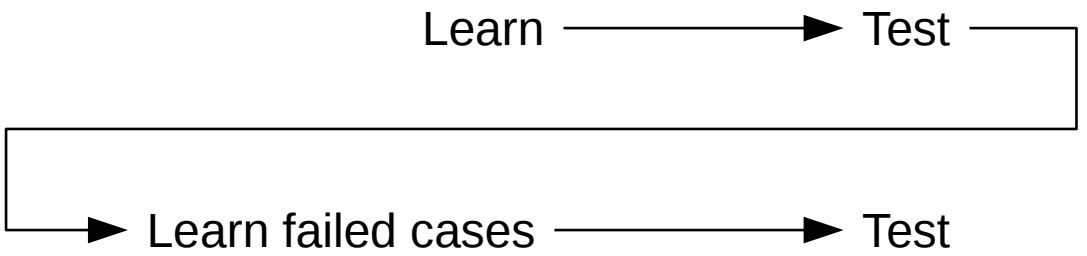
Data: how human learns?



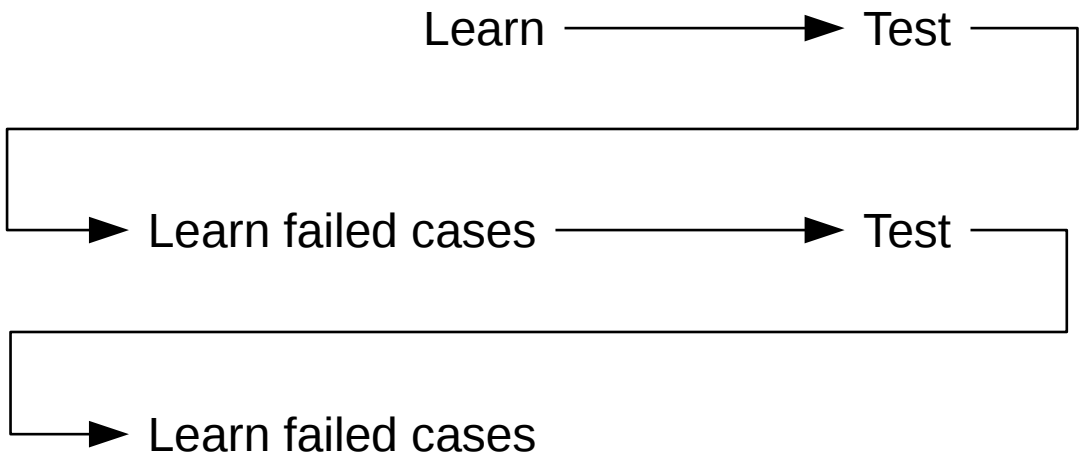
Data: how human learns?



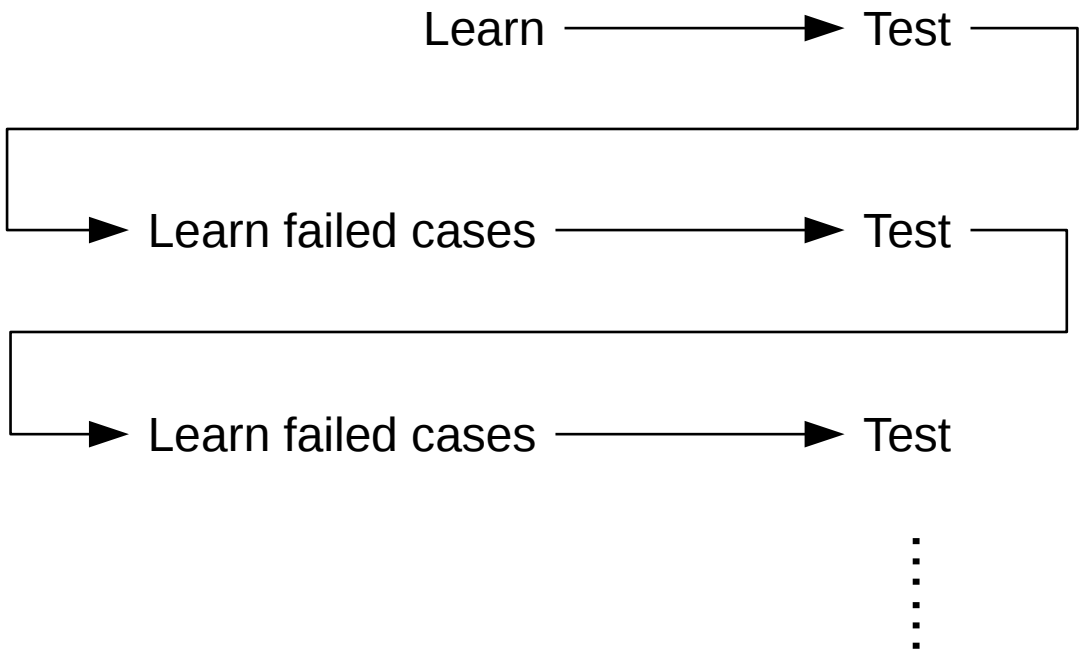
Data: how human learns?



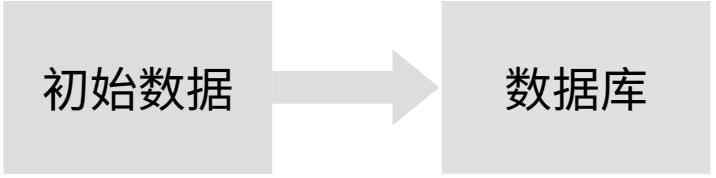
Data: how human learns?



Data: how human learns?

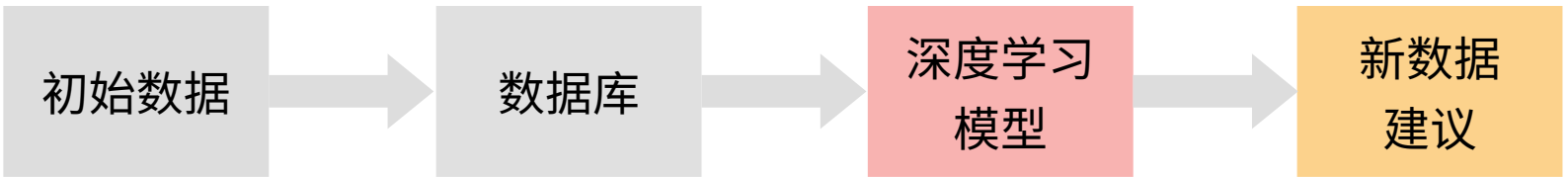


Data: idea of concurrent learning



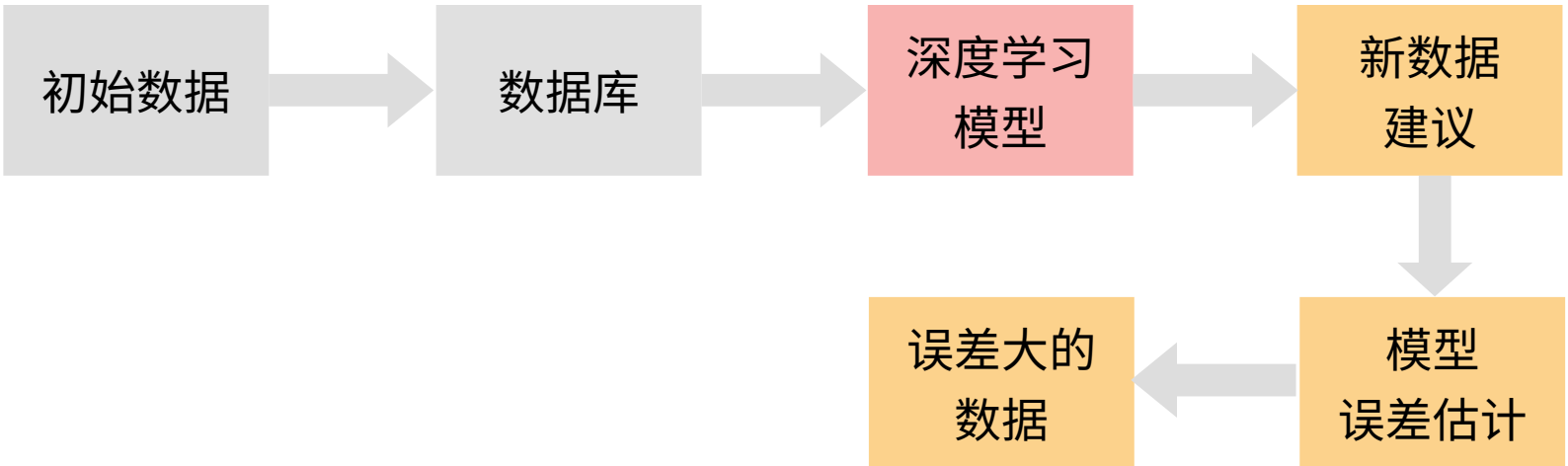
Data: idea of concurrent learning

Data: idea of concurrent learning



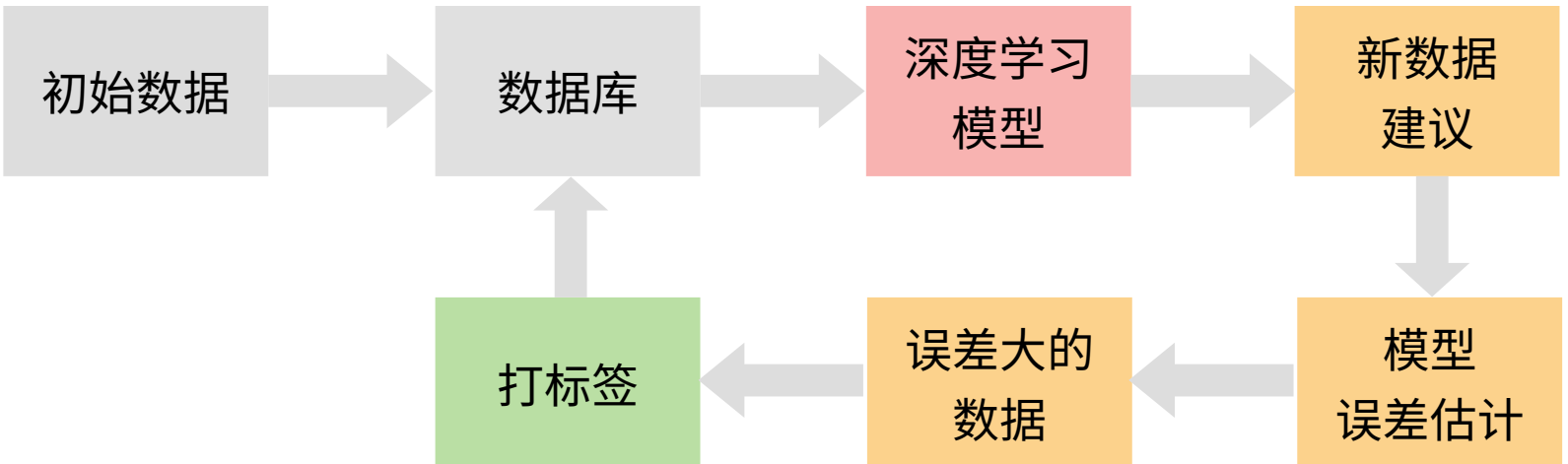
New data generation:
highly depends on application

Data: idea of concurrent learning



First principle comput.
is not needed

Data: idea of concurrent learning



First principle comput.
are needed

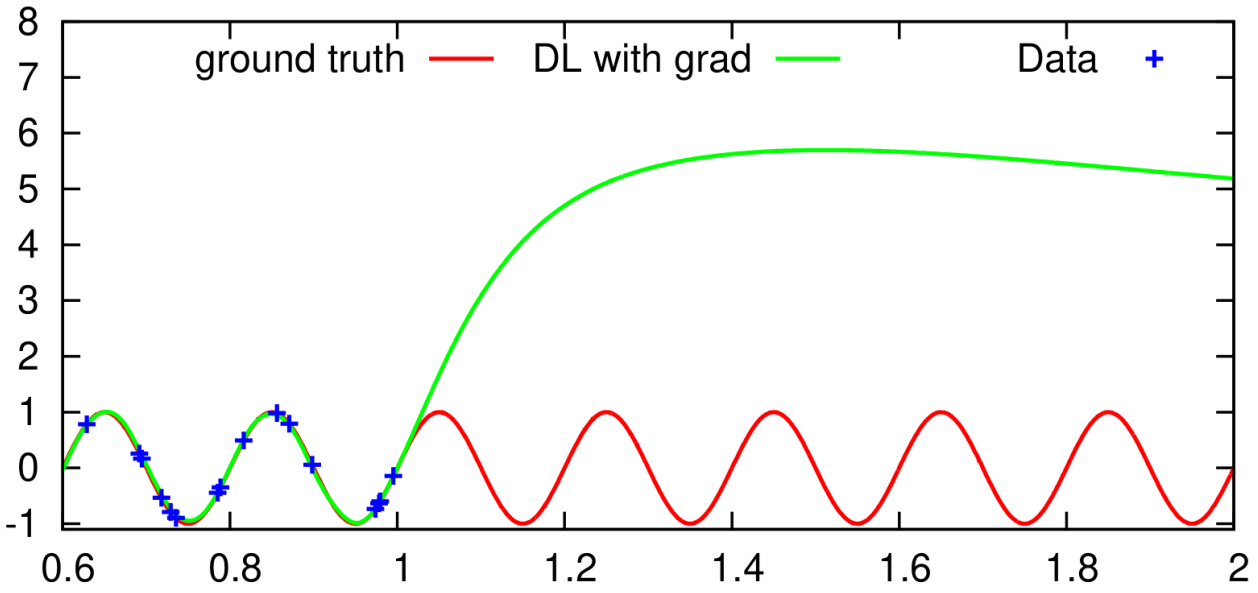
Data: idea of concurrent learning

Model used in different applications?

Accuracy is not expected! Extrapolation is not expected

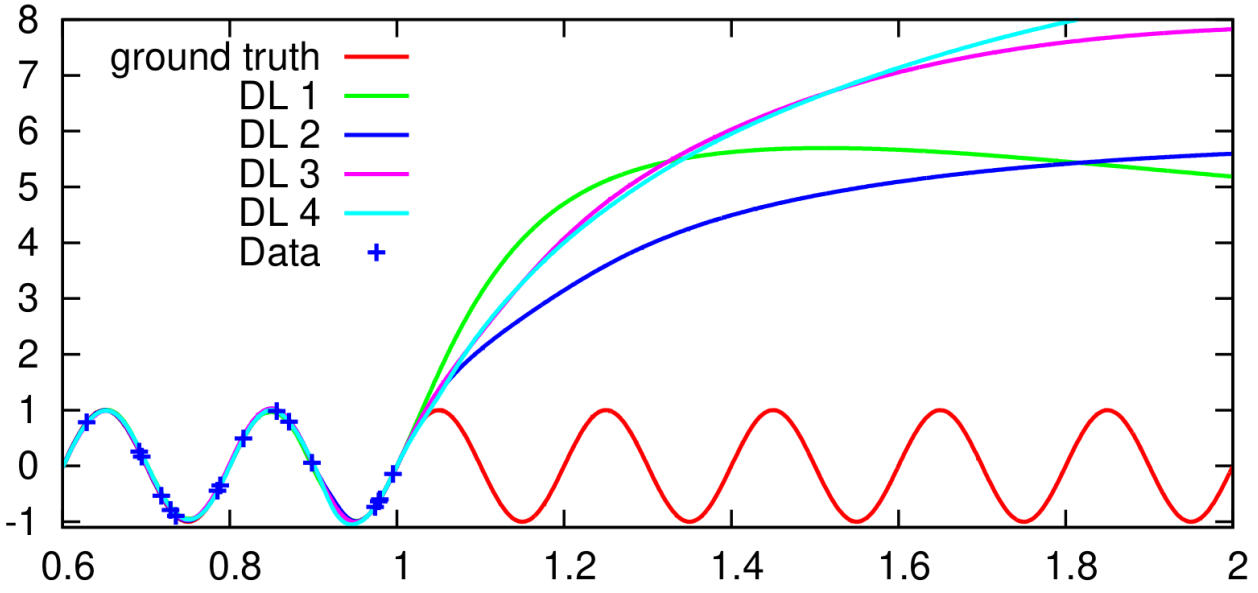
Model used with careful benchmark and error estimator

Toy model: a 1-dimensional function

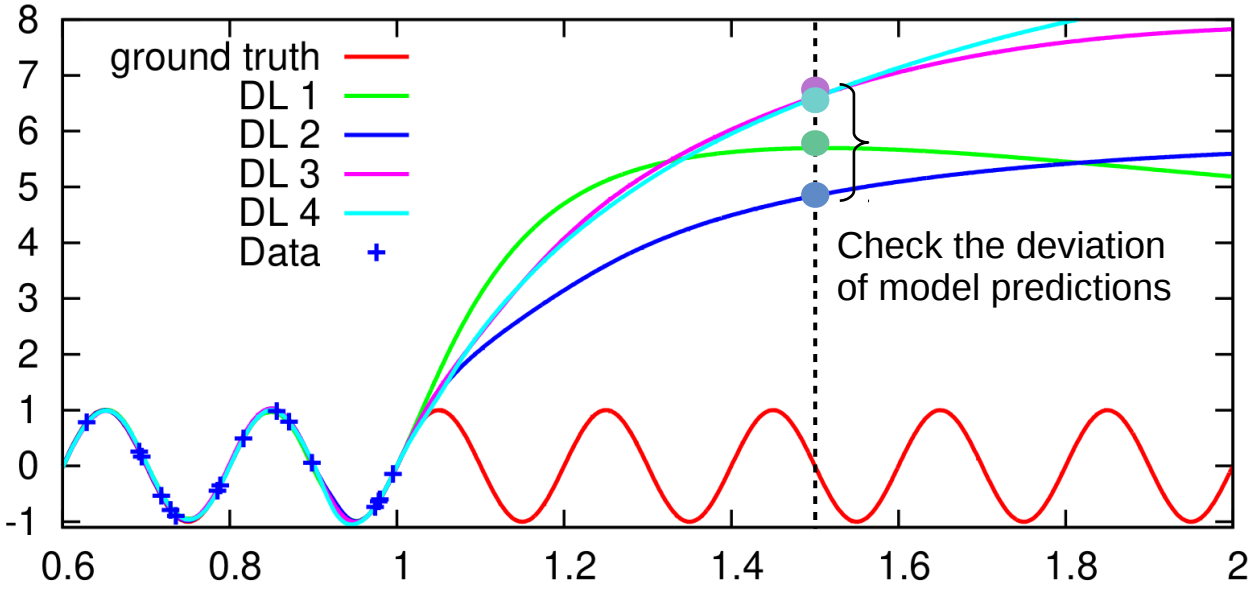


NOT able to “extrapolate” at all

Model deviation: a 1-dimensional function



Model deviation: a 1-dimensional function



Data: concurrent learning for data generation

Step 1

Training:

$$\min_{\omega} \parallel E(\boldsymbol{R},\omega) - E(\boldsymbol{R}) \parallel^2 + \lambda \parallel \boldsymbol{F}(\boldsymbol{R},\omega) - \boldsymbol{F}(\boldsymbol{R}) \parallel^2$$

Random initializations $\omega_1, \omega_2, \ldots,$

The same training procedure

An ensemble of Deep Potentials $E(\boldsymbol{R},\omega_1), E(\boldsymbol{R},\omega_2), \cdots$

Data: concurrent learning for data generation

Step 2

Exploration:

$$\frac{d^2\boldsymbol{R}}{dt^2} = -\nabla E(\boldsymbol{R}, \omega_1)$$

Check model deviation along the trajectories

$$\varepsilon_t = \max_n \sqrt{\langle \| F_n(\boldsymbol{R}_t, \omega) - \langle F_n(\boldsymbol{R}_t, \omega) \rangle_\omega \|^2 \rangle_\omega} \qquad \Delta t, 2\Delta t, 3\Delta t, \ldots$$

Select failed configurations:

$$\{k | \, \varepsilon_{\mathrm{lo}} \leq \varepsilon_{k\Delta t} < \varepsilon_{\mathrm{hi}}\} \quad \Longrightarrow \quad \{\boldsymbol{R}_{k_1\Delta t}, \boldsymbol{R}_{k_2\Delta t}, \cdots\}$$

Data: concurrent learning for data generation

Step 3

Labeling:

For each selected configuration, $\{\boldsymbol{R}_{k_1\Delta t}, \boldsymbol{R}_{k_2\Delta t}, \cdots\}$,

compute $E(\boldsymbol{R}_{k_i\Delta t})$ and $F(\boldsymbol{R}_{k_i\Delta t}) = -\nabla E(\boldsymbol{R}_{k_i\Delta t})$

Extend the data set by $\{(\boldsymbol{R}_{k_1\Delta t}, E(\boldsymbol{R}_{k_1\Delta t})), (\boldsymbol{R}_{k_2\Delta t}, E(\boldsymbol{R}_{k_2\Delta t})), \dots\}$

DP-GEN: Deep potential generator

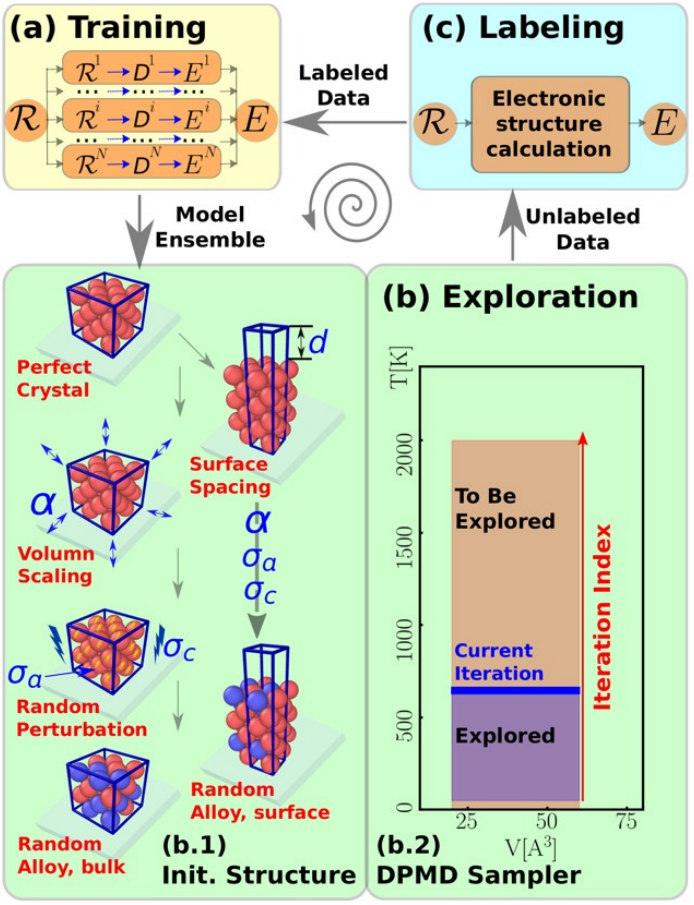
DP-GEN:
Deep Potential Generator

Key step: exploration

MD initial configurations

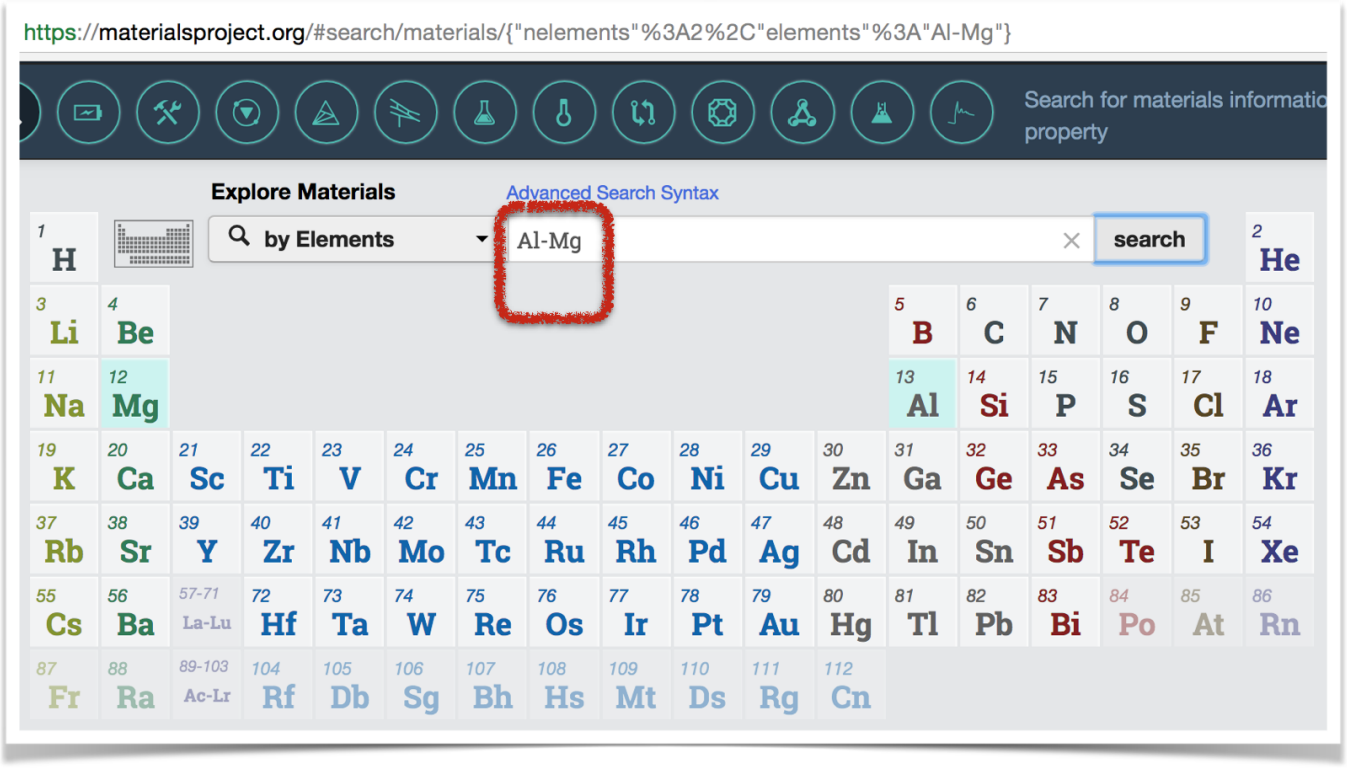
1. scaling
2. perturbation
3. random occupation

MD themodynamic conditions:
scan a user defined range of thermodynamic conditions



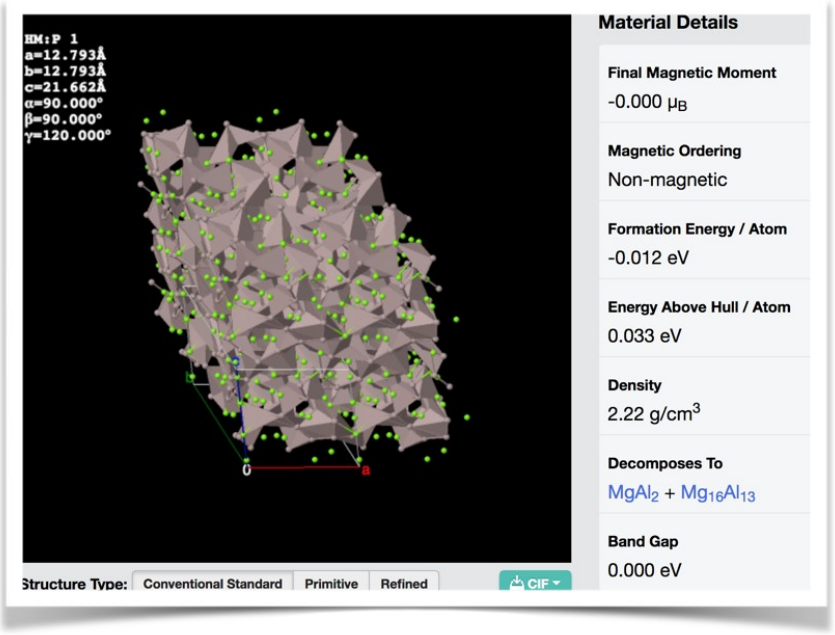
https://github.com/deepmodeling/dpgen

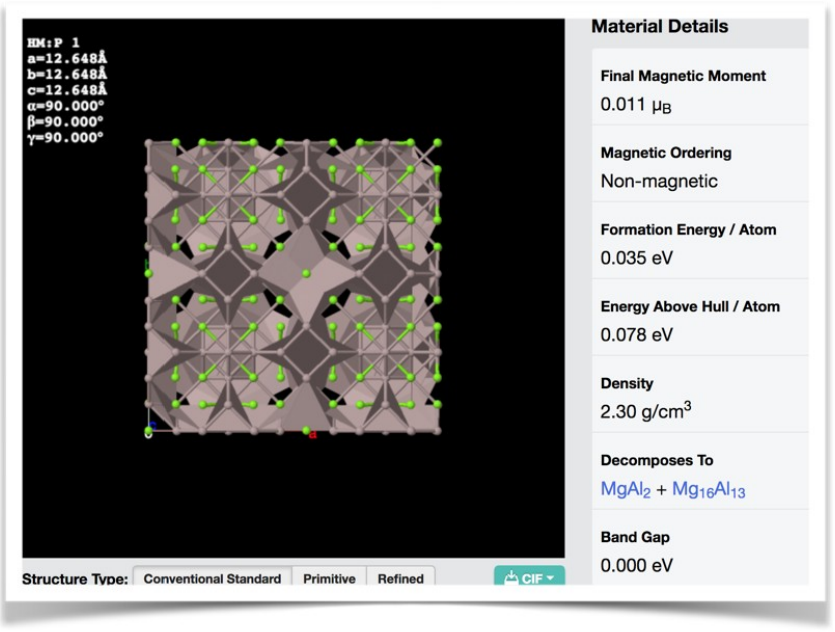
DP-GEN: blind test of alloy properties



DP-GEN: blind test of alloy properties

28 alloy structures, none of them explicitly included in training data

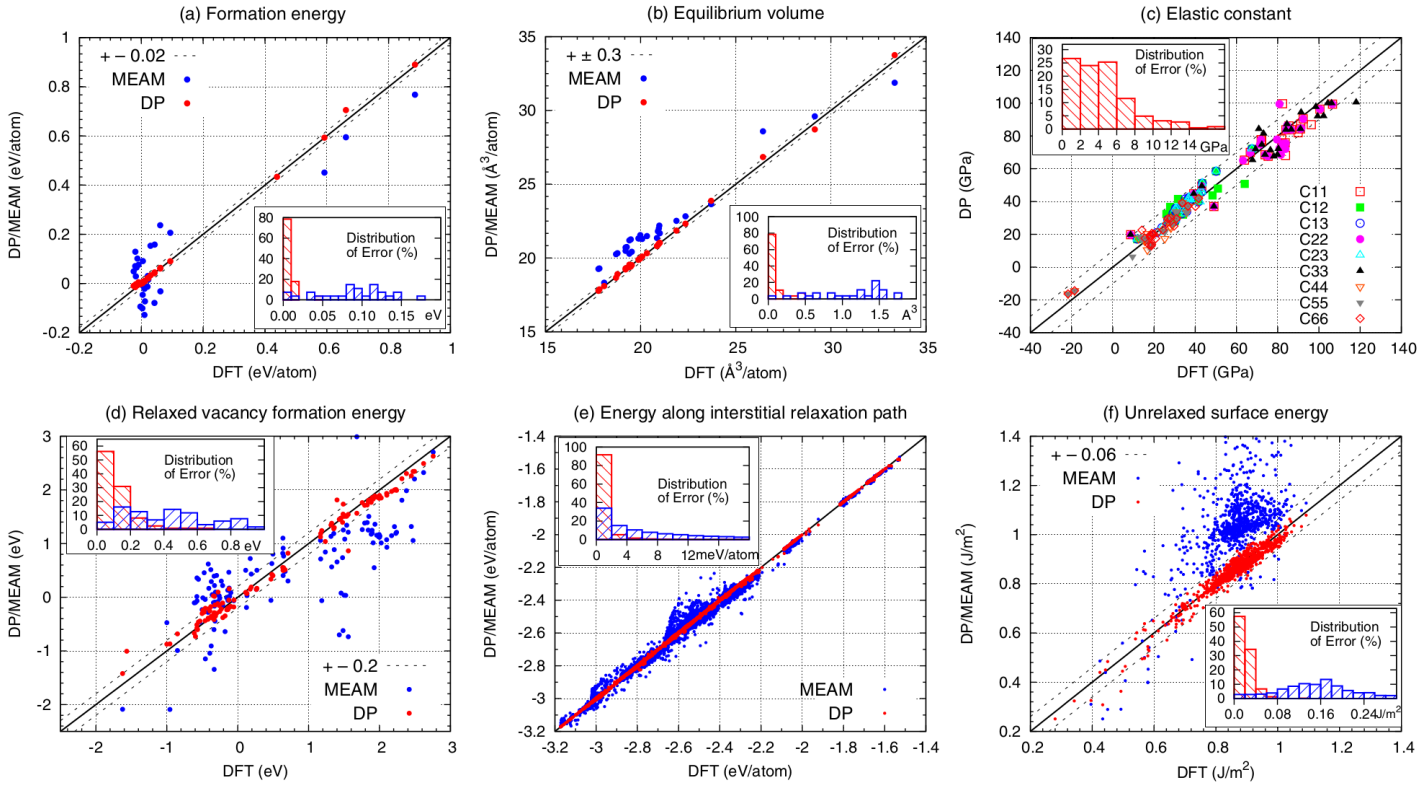




. . .

DP-GEN: blind test of alloy properties

None of the deformed, defect, surface structure is explicitly included in the training data



A brief summary of the results

The coverage of data defined by exploration strategy

Model should be used in the range “covered” by data

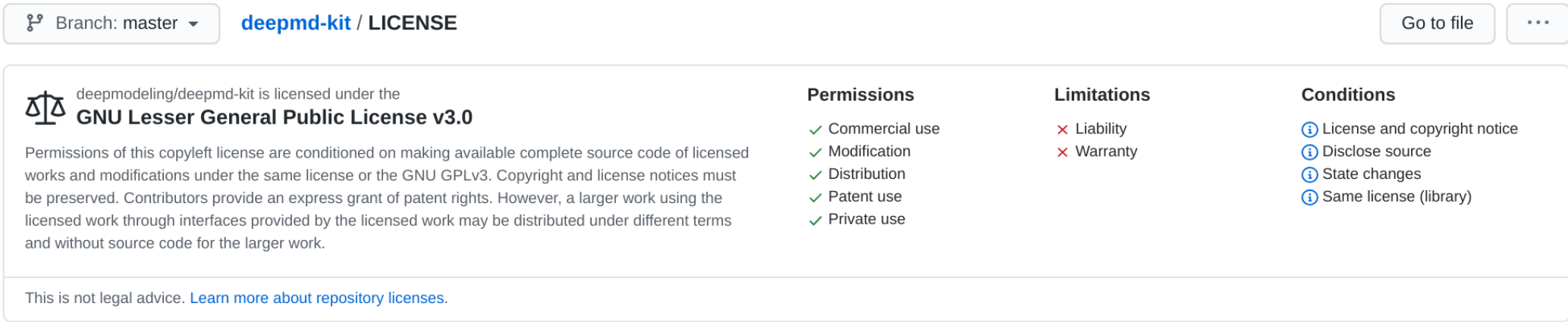
Difficult to check coverage, due to high-dim

Bench mark with first principle result

On-the-fly check with model deviation in MDs

DP and the open source community

Both DeePMD-kit and DP-GEN are released under GNU/LGPLv3

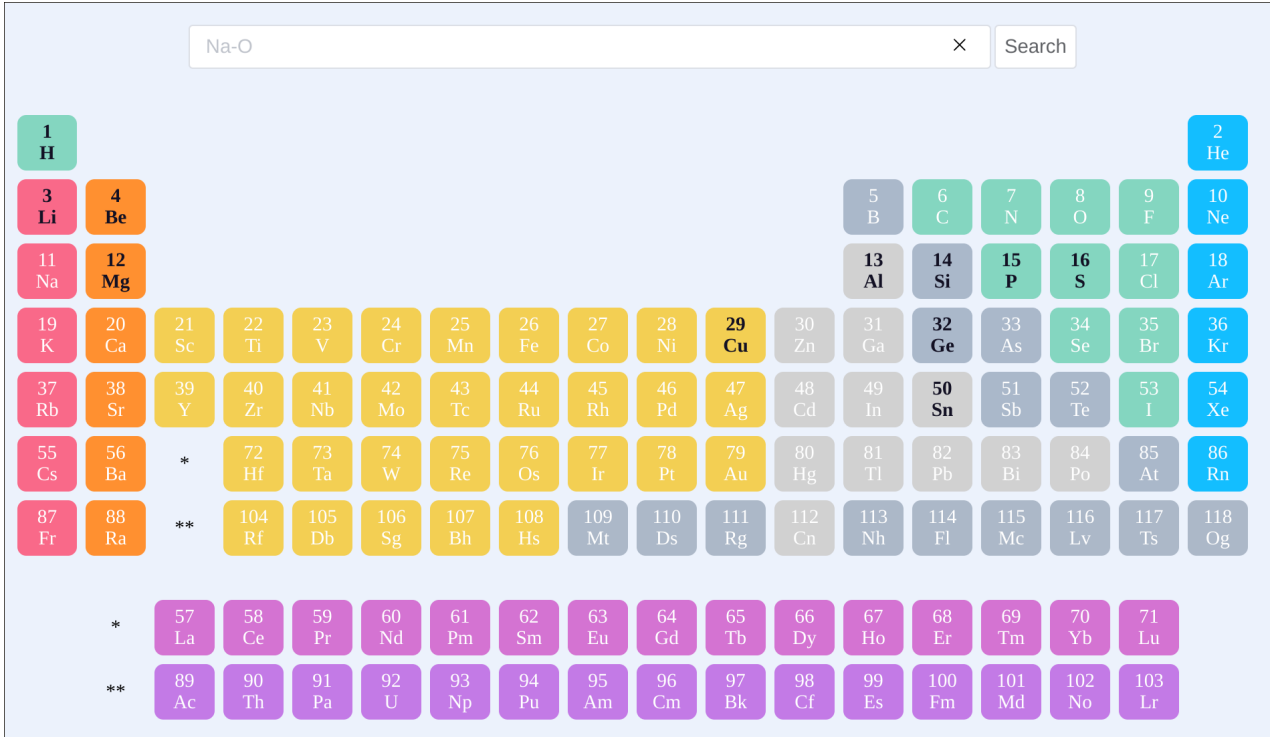


- DeePMD-kit and DP-GEN are free software
- Here “free” means the freedom of accessing the source code, using (incl. commercial use), revising and redistributing the software, not “free of charge”.
- Redistribution based on the software should obey the same license.

The DP-Library project

An open platform for sharing DP models and training data

DP Library



- **Model sharing:**
  Accessibility to published nd well benchmarked models
- **Data sharing:**
  Progressively improving the quality of models
- **Reproducibility**
- **CC license**
  The model and data are essentially owned by the authors.

creative commons

Summary

Deep potential: modeling interatomic potential by deep learning

As accuracy as first principle method and as efficient as empirical force fields

Tensorial properties: IR and Rahman spectra

Deep learning for multiscale modeling

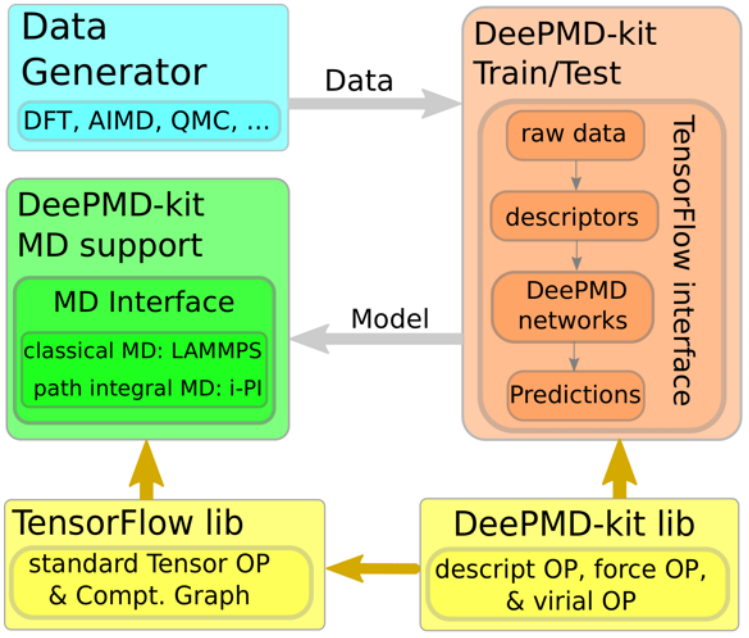
Deep learning for post Hartree methods

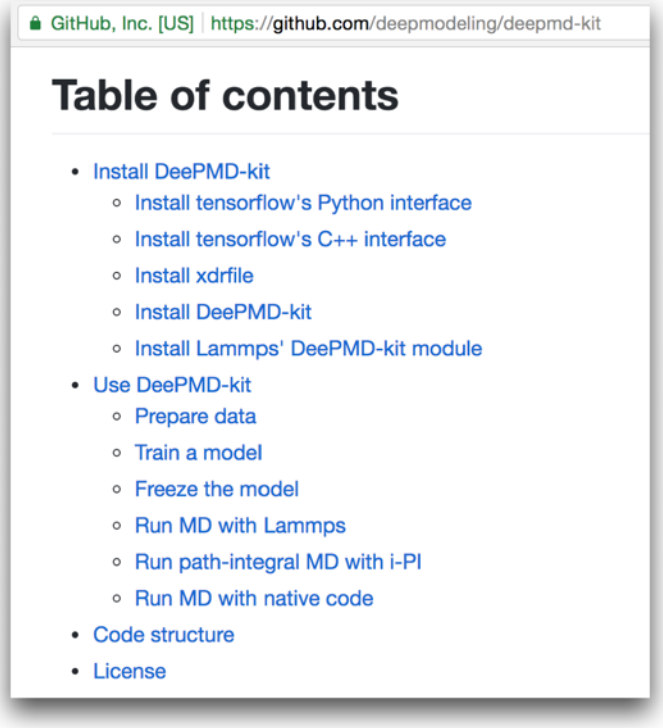
DP meets HPC: 100 million atoms 1ns in 1 day

DP-GEN: achieving uniform accuracy with minimal amount of data

A fully automated data generation scheme

DeePMD-kit: a brief introduction





Free software from https://github.com/deepmodeling/deepmd-kit

DeePMD-kit: a brief introduction

Get the deepmd-kit source code

git clone https://github.com/deepmodeling/deepmd-kit.git

Workflow of DeePMD-kit

Data

```
1. import dpdata
2. dpdata.LabeledSystem('OUTCAR').to('deepmd/npy', 'data', set_size=200)
```

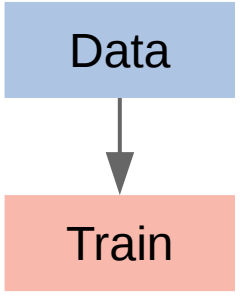
dpdata converts vasp OUTCAR to DeePMD-kit data

DeePMD-kit: a brief introduction

Get the deepmd-kit source code

```
git clone https://github.com/deepmodeling/deepmd-kit.git
```

Workflow of DeePMD-kit




```
1. import dpdata
2. dpdata.LabeledSystem('OUTCAR').to('deepmd/npy', 'data', set_size=200)
```

dpdata converts vasp OUTCAR to DeePMD-kit data

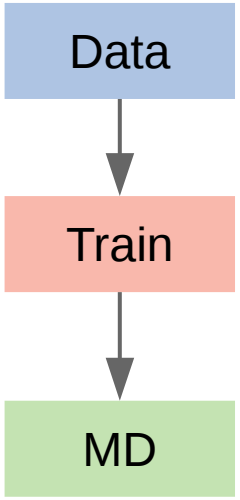
```
1. cd deepmd-kit/examples/water/train/
2. dp train water_se_a.json          # train a model for water
3. dp freeze -o frozen_model.pb      # freeze the model parameters in a file
```

DeePMD-kit: a brief introduction

Get the deepmd-kit source code

git clone https://github.com/deepmodeling/deepmd-kit.git

Workflow of DeePMD-kit




```
1. import dpdata
2. dpdata.LabeledSystem('OUTCAR').to('deepmd/npy', 'data', set_size=200)
```

dpdata converts vasp OUTCAR to DeePMD-kit data

```
1. cd deepmd-kit/examples/water/train/
2. dp train water_se_a.json          # train a model for water
3. dp freeze -o frozen_model.pb      # freeze the model parameters in a file
```

```
1. pair_style    deepmd frozen_model.pb
2. pair_coeff    #              ^^^^^^^^^^^^^^^ file of the model
```

```
1. cd deepmd-kit/examples/water/lmp
2. lmp in.lammps
```

DeePMD-kit: input script

The input script for DeePMD-kit, taking “water_se_a.json” as an example

```
1.  "model": {
2.      "type_map":     ["O", "H"],
3.      "descriptor" :{
4.          "type":             "se_a",
5.          "sel":              [46, 92],
6.          "rcut_smth":    5.80,
7.          "rcut":             6.00,
8.          "_comment":         " that's all"
9.      }
10. }
```

Maximally possible number of neighbors

DeePMD-kit: input script

The input script for DeePMD-kit, taking “water_se_a.json” as an example

```
"model": {
    "type_map":     ["O", "H"],
    "descriptor" :{
        "type":            "se_a",
        "sel":             [46, 92],
        "rcut_smth":   5.80,
        "rcut":            6.00,
        "_comment":        " that's all"
    }
}
```

Maximally possible number of neighbors

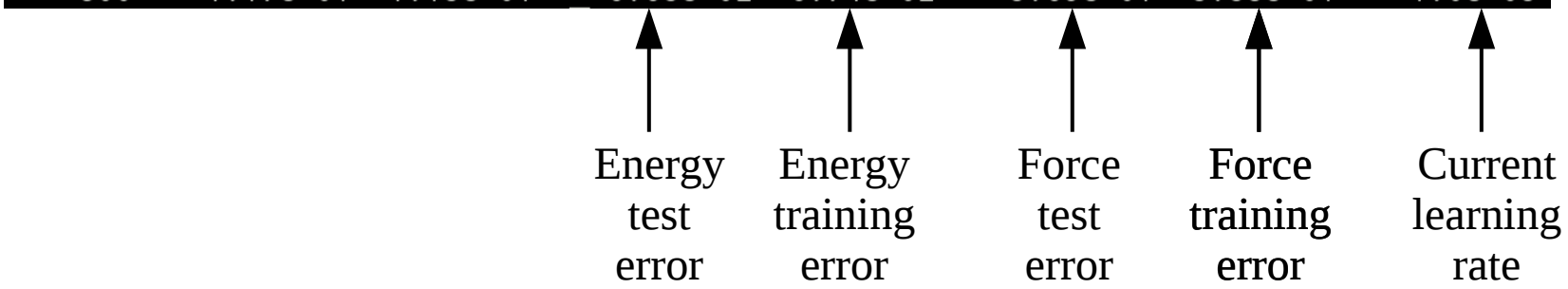
```
"training" : {
    "systems":      ["../data/"],
    "stop_batch":   1000000,
    "batch_size":   "auto"
}
```

The data systems. A system should contain the same number of atoms of the same type

Number of training steps.

DeePMD-kit: lcurve.out

```
[wanghan@han-x1-2 deepmd-kit]$ head  examples/water/train/lcurve.out
# batch      l2_tst    l2_trn    l2_e_tst  l2_e_trn    l2_f_tst  l2_f_trn         lr
      0    3.34e+01  3.31e+01    1.03e+01  1.03e+01    8.42e-01  8.30e-01    1.0e-03
    100    2.69e+01  2.45e+01    1.64e+00  1.64e+00    8.43e-01  7.69e-01    1.0e-03
    200    2.63e+01  2.51e+01    8.70e-02  9.14e-02    8.32e-01  7.95e-01    1.0e-03
    300    2.41e+01  2.52e+01    4.34e-02  4.21e-02    7.63e-01  7.96e-01    1.0e-03
    400    2.24e+01  2.06e+01    3.92e-01  3.90e-01    7.09e-01  6.51e-01    1.0e-03
    500    2.01e+01  1.80e+01    1.17e-02  1.61e-02    6.36e-01  5.69e-01    1.0e-03
    600    1.65e+01  1.68e+01    4.45e-02  5.10e-02    5.23e-01  5.32e-01    1.0e-03
    700    1.38e+01  1.34e+01    1.27e-01  1.28e-01    4.35e-01  4.24e-01    1.0e-03
    800    1.17e+01  1.13e+01    6.65e-02  6.74e-02    3.69e-01  3.58e-01    1.0e-03
```



DeePMD-kit: training data and test data

ls ../data

```
(py3.6-tf1.8) [wanghan@han-x1-2 train]$ ls ../data/
set.000  set.001  set.002  set.003  type.raw
```

Training data

test data

DeePMD-kit: training data and test data

ls ../data

```
(py3.6-tf1.8) [wanghan@han-x1-2 train]$ ls ../data/
set.000  set.001  set.002  set.003  type.raw
```

Training data

test data

dp test -m frozen_model.pb -s ../data -n 200

```
# ---------------output of dp test---------------
# testing system : ../data
# number of test data : 100
Energy L2err        : 1.777362e+01 eV
Energy L2err/Natoms : 9.257096e-02 eV
Force  L2err        : 1.976301e-01 eV/A
Virial L2err        : 2.662693e+01 eV
Virial L2err/Natoms : 1.386819e-01 eV
# -----------------------------------------------
```

Errors of the model tested on set.003

DeePMD-kit: model deviation

Check the quality of the model on the fly

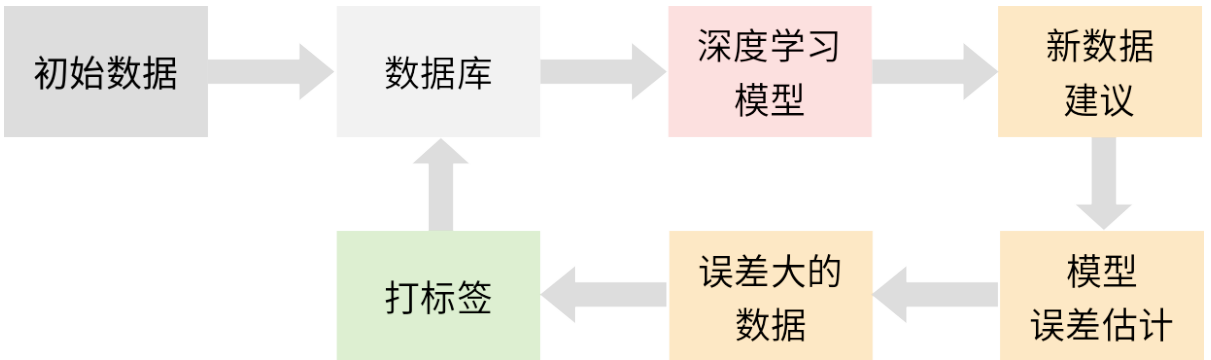
```
pair_style    deepmd  graph_0.pb  graph_1.pb  graph_2.pb  graph_3.pb   out_file  md.out   out_freq  10
pair_coeff
```

# step	max_devi_e	min_devi_e	avg_devi_e	max_devi_f	min_devi_f	avg_devi_f
0	9.376648e-04	5.815220e-04	7.408408e-04	7.915502e-03	5.364027e-03	6.459966e-03
10	6.374548e-03	4.748968e-04	2.960457e-03	4.940441e-02	1.280846e-02	2.951274e-02
20	8.521549e-03	4.287206e-04	3.479513e-03	5.559978e-02	1.424634e-02	3.022141e-02
30	1.327014e-02	5.384389e-04	3.685414e-03	7.789596e-02	1.417084e-02	3.589276e-02
40	1.133173e-02	8.557426e-04	3.458947e-03	6.179347e-02	1.089575e-02	3.077951e-02
50	7.103331e-03	7.630305e-04	2.718725e-03	4.787257e-02	1.385170e-02	2.976341e-02
60	5.557966e-03	4.172685e-04	2.562101e-03	4.131747e-02	1.257815e-02	2.604306e-02
70	1.127818e-02	5.224720e-04	3.108563e-03	5.720856e-02	1.388272e-02	3.043037e-02
80	7.570629e-03	4.844596e-04	2.950598e-03	6.521619e-02	1.552110e-02	3.046030e-02
90	6.818869e-03	5.327105e-04	3.299618e-03	5.432117e-02	1.600004e-02	3.172750e-02
100	7.713344e-03	2.666449e-04	3.153897e-03	5.889230e-02	1.224567e-02	3.158061e-02

Deprecated, printed for compatibility reason

Maximal model deviation of force predictions

DP-GEN: deep potential generator




```
$ pip install dpgen
$ dpgen run param.json machine.json
```

Workflow of dpgen

Computational resources

DP-GEN: exploration and model deviation

- The dpgen iterations are defined by model_devi_jobs
- Each item specifies the explorations in an dpgen iteration

```
"model_devi_jobs":  [
    {
        "ensemble": "npt",
        "nsteps": 1000,
        "traj_freq": 10,
        "press": [1.0,10.0,100.0,1000.0,5000.0,10000.0,20000.0,50000.0],
        "temps": [50,132.0,198.0,264.0],
        "sys_idx": [0,8,16]
    },
```

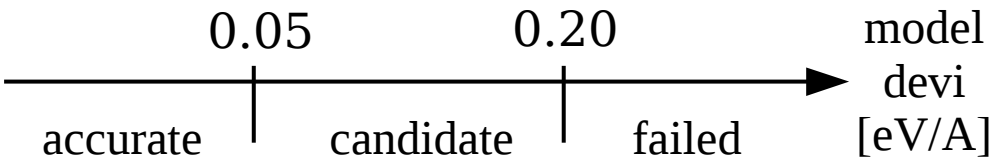
DP-GEN: exploration and model deviation

- The dpgen iterations are defined by model_devi_jobs
- Each item specifies the explorations in an dpgen iteration

```
"model_devi_jobs":  [
    {
        "ensemble": "npt",
        "nsteps": 1000,
        "traj_freq": 10,
        "press": [1.0,10.0,100.0,1000.0,5000.0,10000.0,20000.0,50000.0],
        "temps": [50,132.0,198.0,264.0],
        "sys_idx": [0,8,16]
    },
```

- Explored configurations are selected by model deviation according to 2 levels

```
"model_devi_f_trust_lo":    0.05,
"model_devi_f_trust_hi":    0.20,
```



DP-GEN: labeling and training

Taking VASP as an example

```
"fp_style":          "vasp",
"fp_task_max":       300,
"fp_task_min":       5,
"fp_incar":          "/data1/yfb222333/2_dpgen_gpu_multi/INCAR_metal_scf_gpu",
"fp_pp_path":        "/data1/yfb222333/2_dpgen_gpu_multi/POTCAR-Al",
"fp_pp_files":       ["POTCAR"],
```

Supported packages: Quantum Expresso, Gaussian, CP2K, SIESTA, PWMAT

Training: by DeePMD-kit, user provided default training script.

```
"default_training_param" :
```

Reference

- Deep potential and DeePMD-kit
  - Physical Review Letters, 120, 143001 (2018).
  - NIPS pp. 4436–4446
  - Comput. Phys. Comm., 228, 178-184, (2018).
- DP-GEN
  - Physical Review Materials, 3, 023804 (2019).
  - Comput. Phys. Comm., Aug.107206 (2020).
- DP for tensorial
  - Accepted by PRB, arXiv:1906.11434
  - Phys. Chem. Chem. Phys. 22, 10592-10602 (2020)
- DP and high-performance computing
  - arXiv: 2005.00223
  - arXiv: 2004.11658
- Deep learning for post Hartree Fock
  - arXiv: 2005.00169
- DP for multiscale modeling
  - J. Chem. Phys., 148, 124113 (2018).
  - J. Chem. Phys., 149, 034101 (2018).
  - J. Chem. Phys., 149, 154107 (2018).

Applications:

- H2O-TiO2 surf.: M.F.C.Andrade, Chem. Sci. (2020).
- Combustion: arXiv:1911.12252
- Irradiation damage: APL 114 (24), 244101, 2019
- Warm dense matter: arXiv:1909.00231
- Phase transition of Si: Phys. Rev. Lett. 121, 265701, (2018)
- Excited state: JPCL 2018, 9, 23, 6702-6708, (2018)
- Phase transition memory: IEEE Electron Device Letters (2020)

Acknowledgment

Collaborators

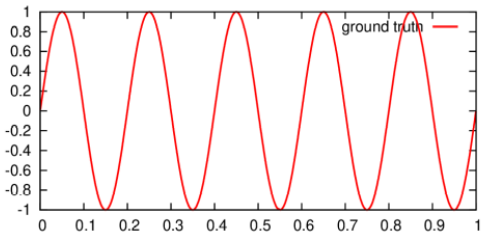
- Linfeng Zhang, Jiequn Han, Yixiao Chen, Roberto Car, Weinan E (Princeton Univ.)
- De-Ye Lin (IAPCM)
- Yuzhi Zhang, Mohan Chen (Peking Univ.)
- Jinzhe Zeng, Tong Zhu (East China Normal Univ.)
- Wissam A Saidi (Univ. of Pittsburgh)
- Many others who contribute to DeePMD-kit and DP-GEN

Funding support

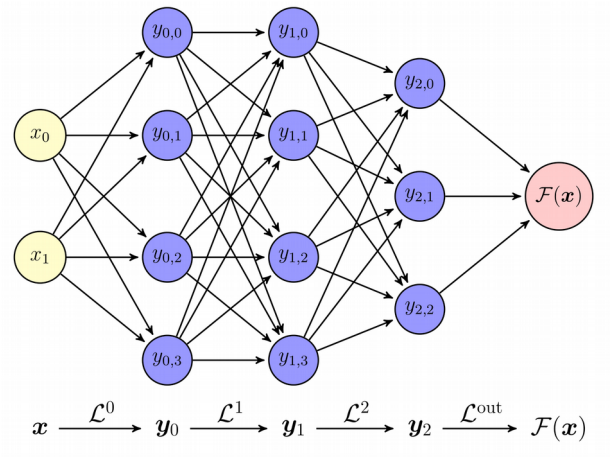
- NSFC 11501039, 11871110, U1430237 and 91530322.
- National Key Research and Development Program of China 2016YFB0201200.
- BAAI
- Beijing Institute of Big Data Research & Tiger@PU

Thank you! Questions?

Deep learning interatomic potential: setup



Learn the result of a one-dim func.



Target: A one-dimensional function in [0,1)

$$f(x) = \sin(10\pi x)$$

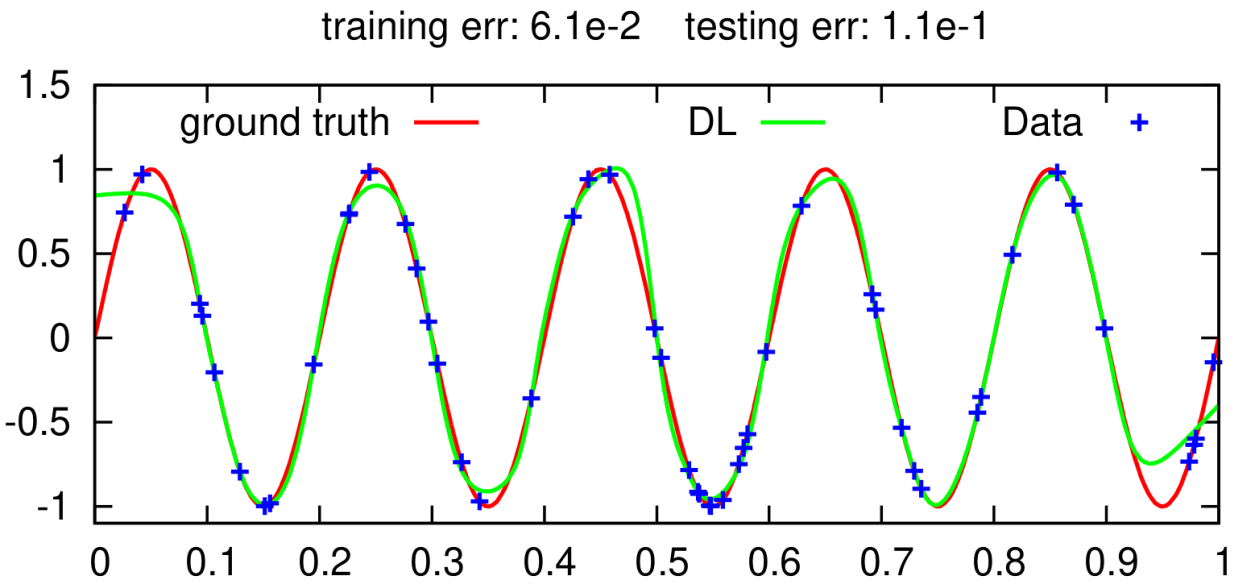
Model: $\mathcal{F}(x, w)$ DNN: 50 x 50 x 50 x 50

Data: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

$$y_i = f(x_i)$$

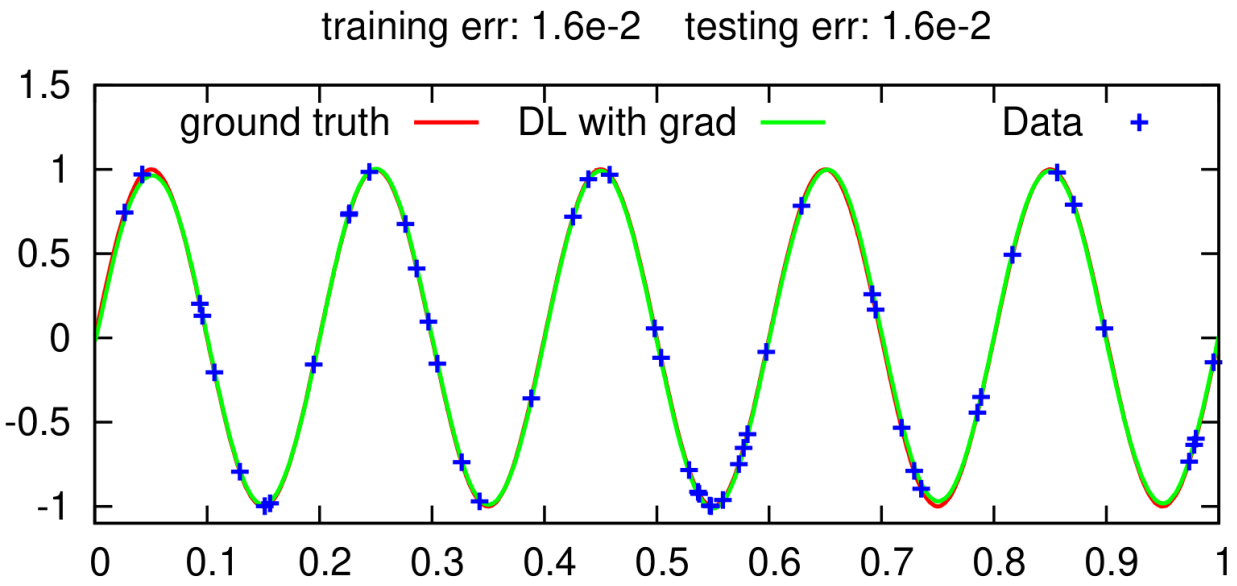
Training: $\min_w \parallel \mathcal{F}(x_i, w) - y_i \parallel^2$

Toy model: a 1-dimensional function



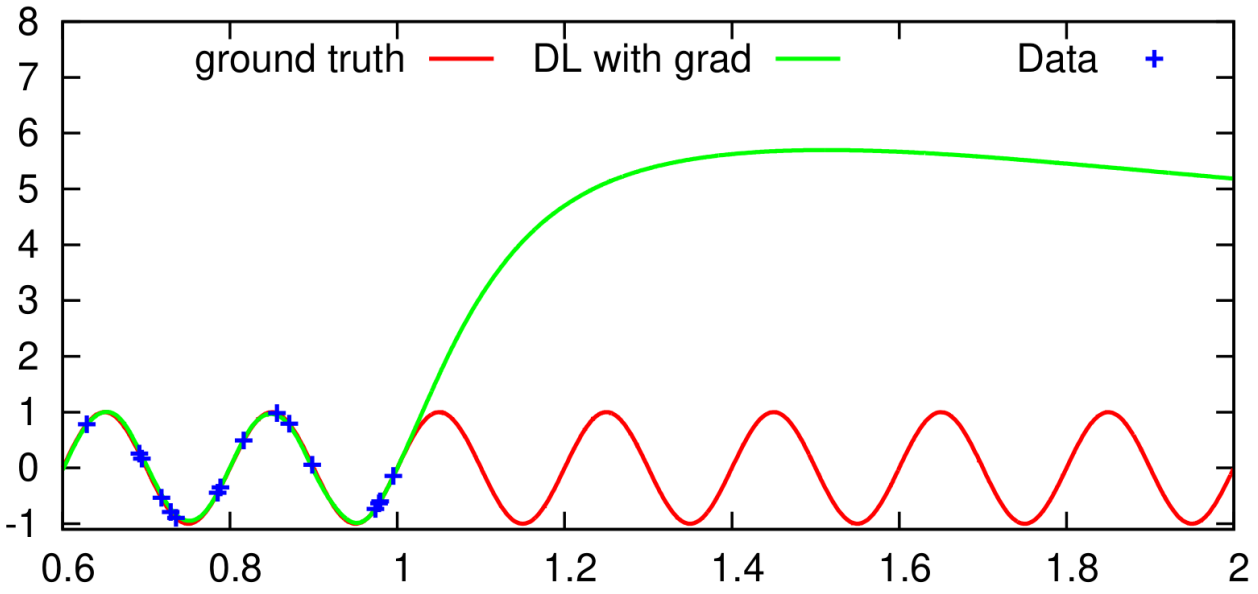
Loss $\| \mathcal{F}(x, w) - f(x) \|^2$

Toy model: a 1-dimensional function



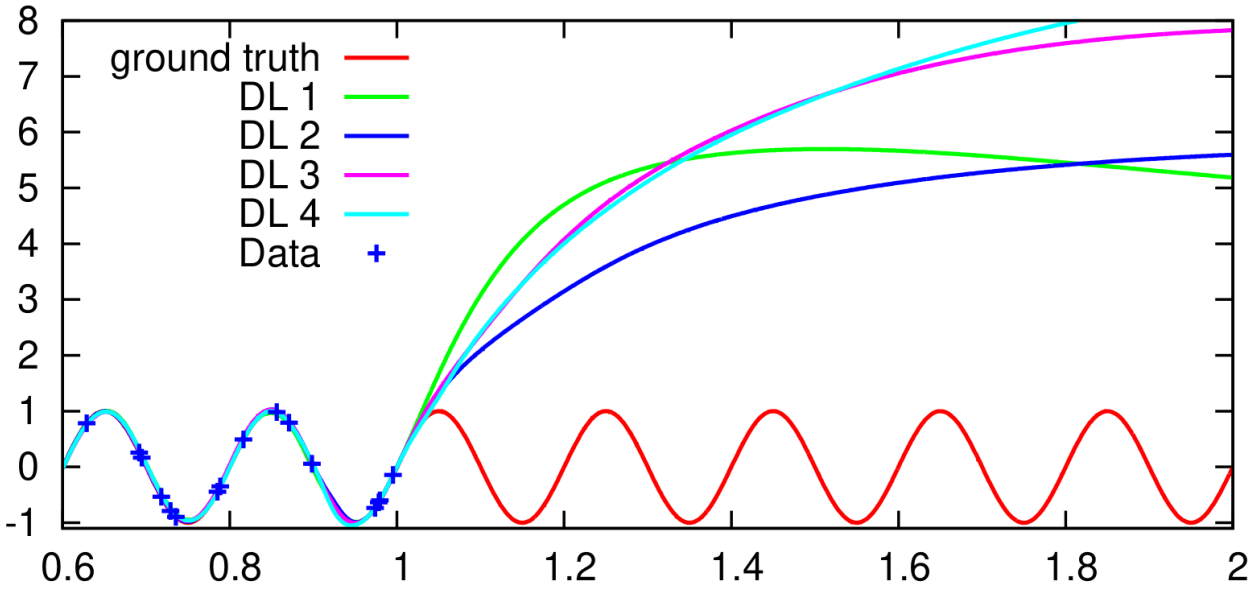
Loss $\| \mathcal{F}(x,w) - f(x) \|^2 + \lambda \| \partial_x F(x,w) - \partial_x f(x) \|^2$

Toy model: a 1-dimensional function

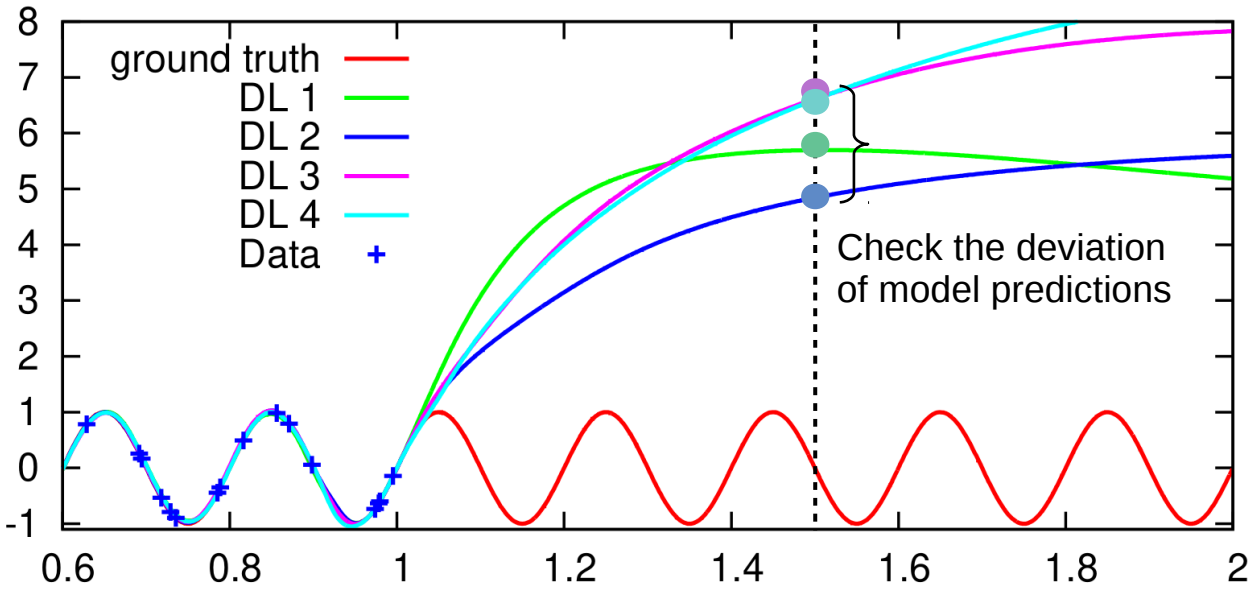


NOT able to “extrapolate” at all

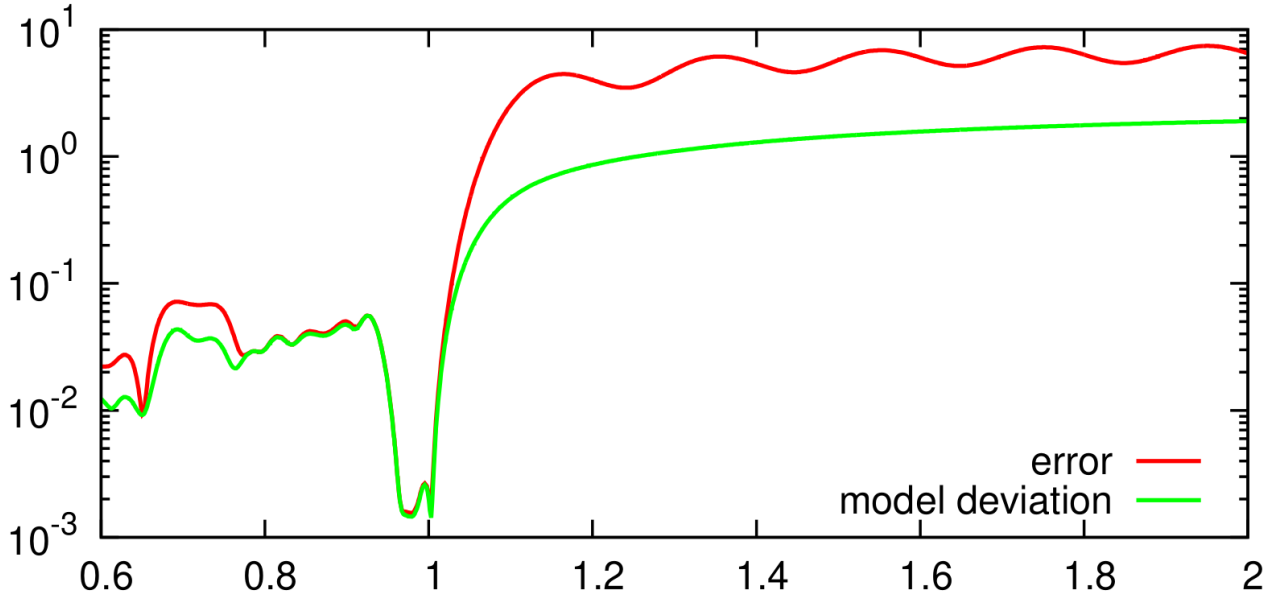
Toy model: a 1-dimensional function



Toy model: a 1-dimensional function



Toy model: a 1-dimensional function

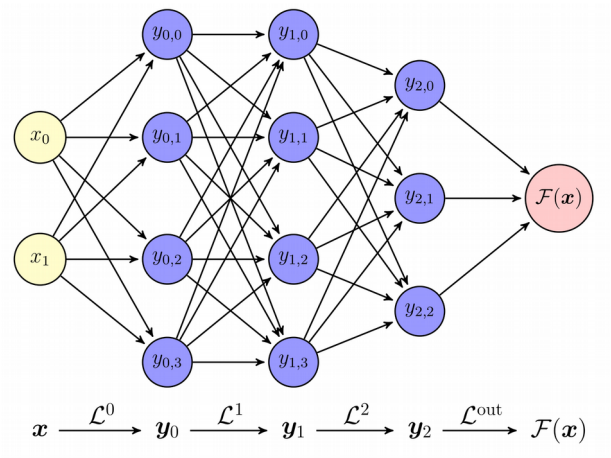


Prescription: model used with the error indicator

Deep learning interatomic potential

$$i\hbar\frac{\partial}{\partial t}|\Psi\rangle = H|\Psi\rangle$$

Learn the result of quantum mech.



Target: Quantum mechanical energy

$E(\boldsymbol{R})$, $\boldsymbol{R} = \{\boldsymbol{r}_i\}$ dimension: 3N

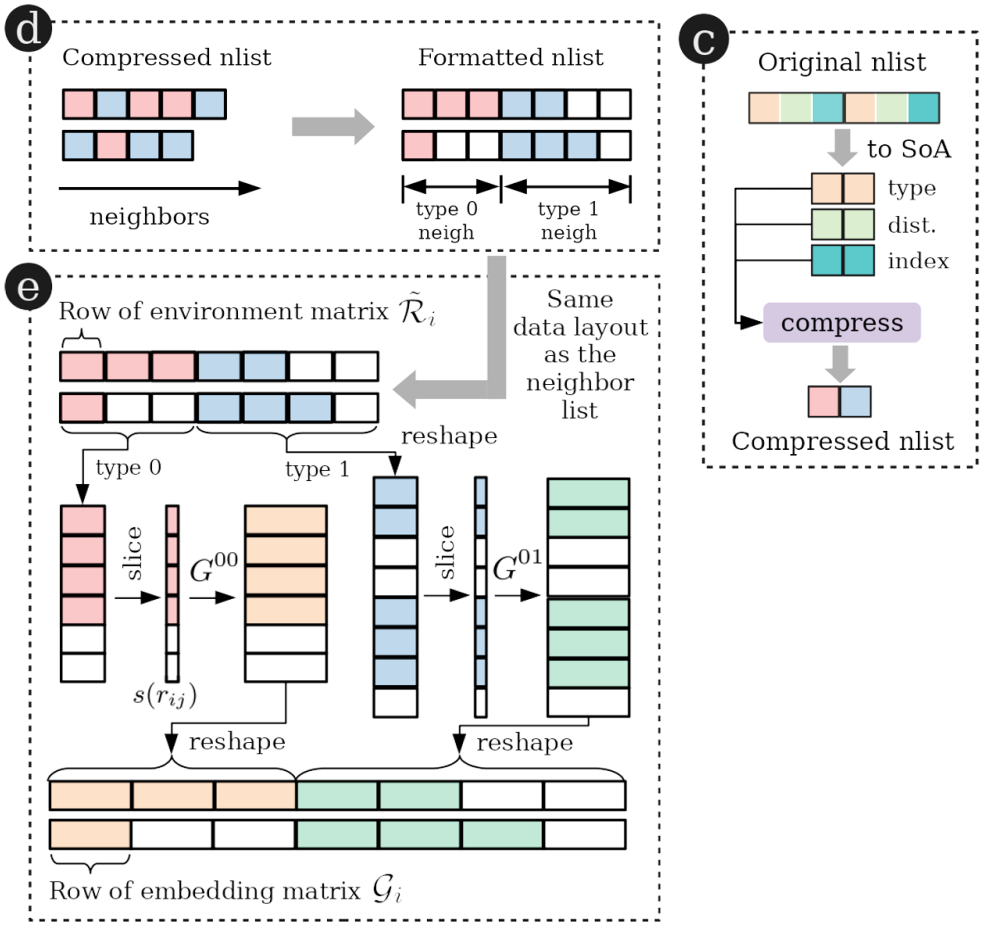
Model: $E(\boldsymbol{R}, \omega)$

Data: $\{(\boldsymbol{R}_0, E_0), (\boldsymbol{R}_1, E_1), (\boldsymbol{R}_2, E_2), \dots\}$

Given $\boldsymbol{R}_i$, $E_i = E(\boldsymbol{R}_i)$ computed by quantum mechanical method

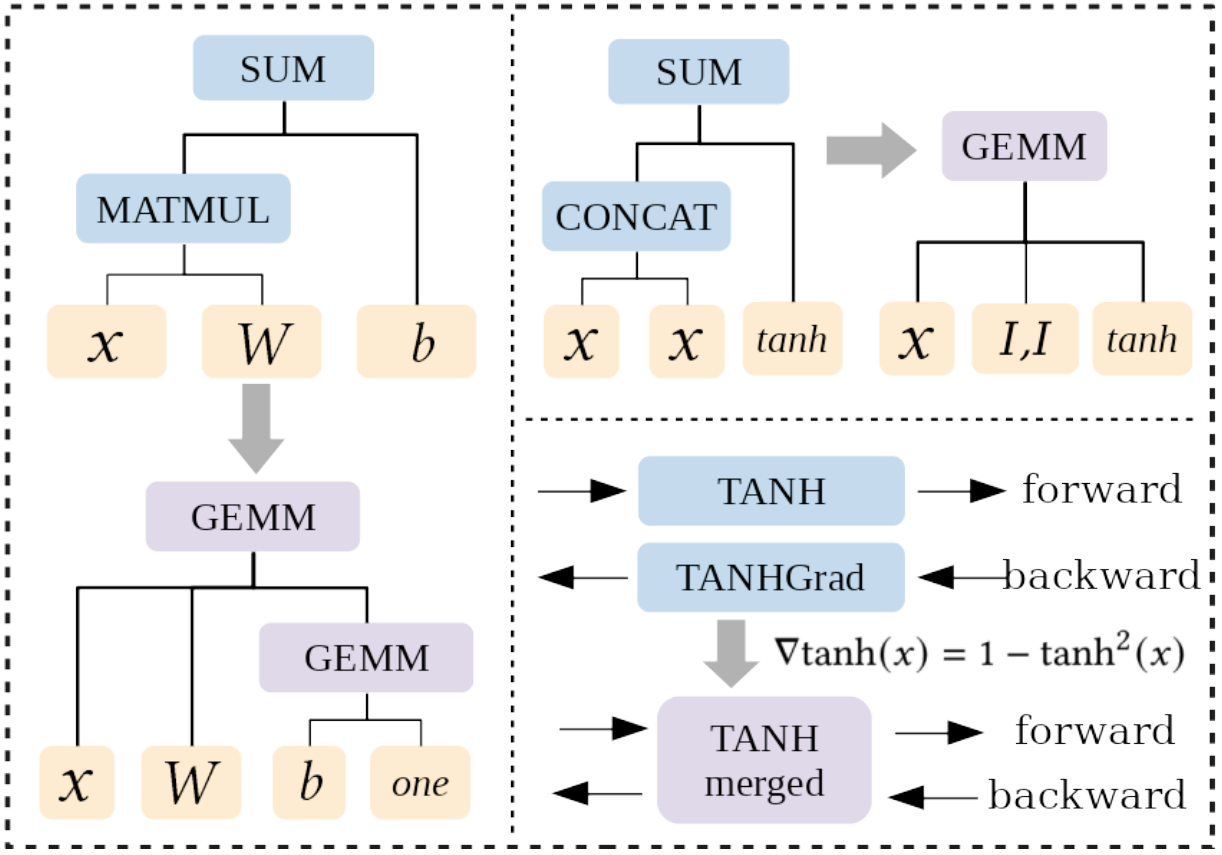
Training: $\min_{\omega} \parallel E(\boldsymbol{R}_i, \omega) - E_i \parallel^2$

DeePMD-kit: optimization for super computers



- 计算环境矩阵$\mathcal{R}_i$时，不同类型邻居原子的操作不同
- 对邻居按类型排序增加计算粒度
- 对邻居表进行压缩实现高效排序

DeePMD-kit: optimization for super computers



TensorFlow 默认计算图优化

计算效率提升 39% 30% 37%