DeePMD调研

范翔宇 ad2018@mail.ustc.edu.cn

DeePMD简介

MD简介

MD全称molecular dynamics，即分子动力学。分子动力学是一套分子模拟方法，该方法主要是依靠计算机来模拟分子、原子体系的运动，是一种多体模拟方法。通过对分子、原子在一定时间内运动状态的模拟，从而以动态观点考察系统随时间演化的行为。通常，分子、原子的轨迹是通过数值求解牛顿运动方程得到，势能(或其对笛卡尔坐标的一阶偏导数，即力)通常可以由分子间相互作用势能函数、力场、全始计算给出。其基本过程为：1) 设置研究对象组成原子的初始位置和速度；2) 计算每个原子受到的合力，并基于牛顿第二定律计算原子的加速度；3) 计算原子下一时刻的速度；4) 计算原子下一时刻的位置；5) 循环2) - 4)的过程，得到一系列时刻原子的位置和速度；6) 基于位置和速度信息得到描述对象性质和行为的物理量。在整个计算过程中，最重要的是如何计算原子间相互作用力。而力是势能的一阶导，所以我们致力于求解势能E。下面介绍求解E的两种方法。

第一性原理(ab initio molecular dynamics即AIMD)

具体细节过于专业，在此不做阐述，可以理解为一个黑盒，我们向黑盒输入原子坐标等信息，黑盒会输出力和势能。该方法准确但十分昂贵。

经验力场

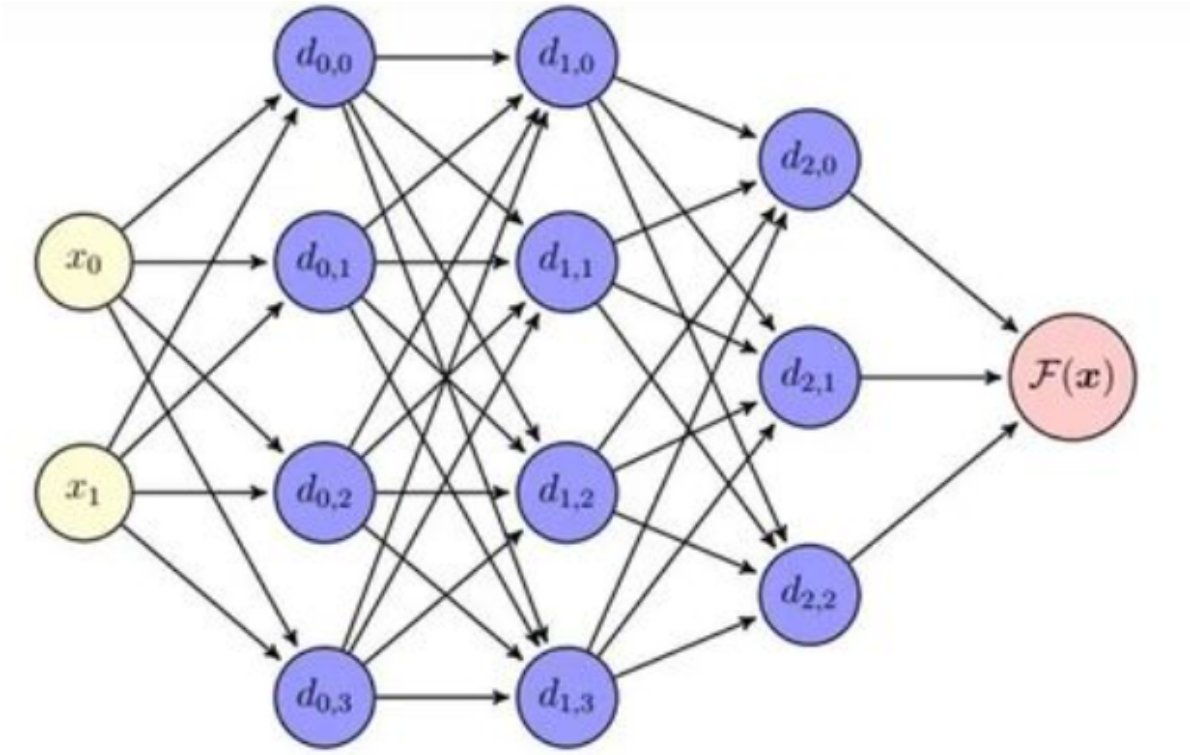
根据物理直觉模拟一个势函数，如Lennard-Jones potential：

$$V_{I,J}=4\epsilon[(\frac{\sigma}{r})^{12}-(\frac{\sigma}{r})^{6}]=\epsilon[(\frac{r_m}{r})^{12}-2(\frac{r_m}{r})^{6}]$$

但是这种势能函数的复杂度较灵活，有时并不能准确计算势能但相对计算迅速。

Deep Potential

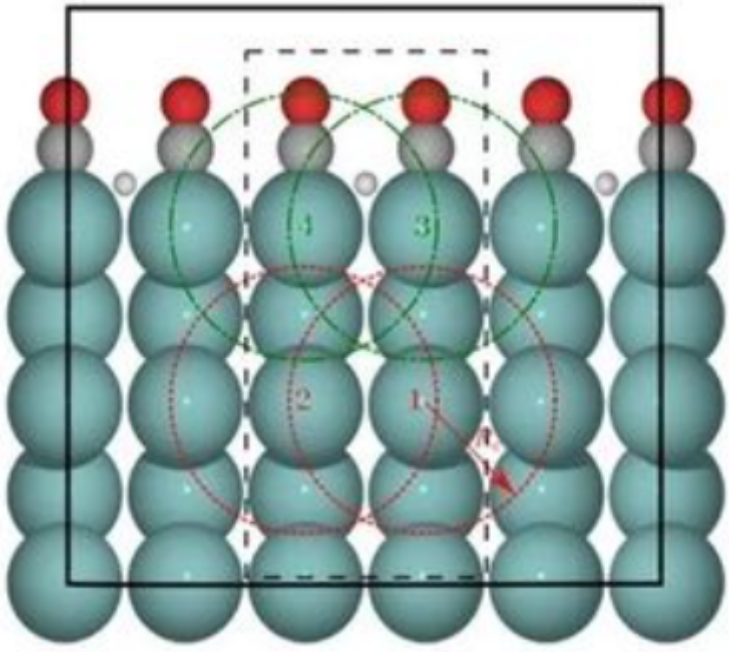
所以为了平衡性能和代价，Deep Potential应运而生，用神经网络来训练。求解势函数的过程即求解势能面的过程，进而转化为求解神经网络中这些参数的过程，即参数拟合问题。



对通过第一性原理计算的精确数据进行拟合，我们希望得到势函数的系数进而得到具体函数。

$$\left\{(\tilde{\mathbf{R}}_1, E_1), (\tilde{\mathbf{R}}_2, E_2), \ldots\right\} \xrightarrow{\text{feature map and regression}} E = \sum_{i=1}^{N} \hat{E}_i(\mathbf{G}_i(\tilde{\mathbf{R}}^{(\mathrm{loc})}))$$

可以认为整体能量是分子能量之和，即$E = \Sigma_i E_i$，而分子能量又取决于邻居间的相互作用。



当然，我们还要所求势函数要有良好的可扩展性，如我们训练计算时只有30个原子，但是我们的模型应该可以准确应用于300000个原子。除此之外，我们要保证势能面的平移、旋转和置换的对称性。对称性可以理解为我进行上述操作时原子的能量是不变的，而坐标其实不能直接输入神经网络，因此我们要将其转化为具有一定对称性的描述符。

首先将坐标转化为距离矩阵：

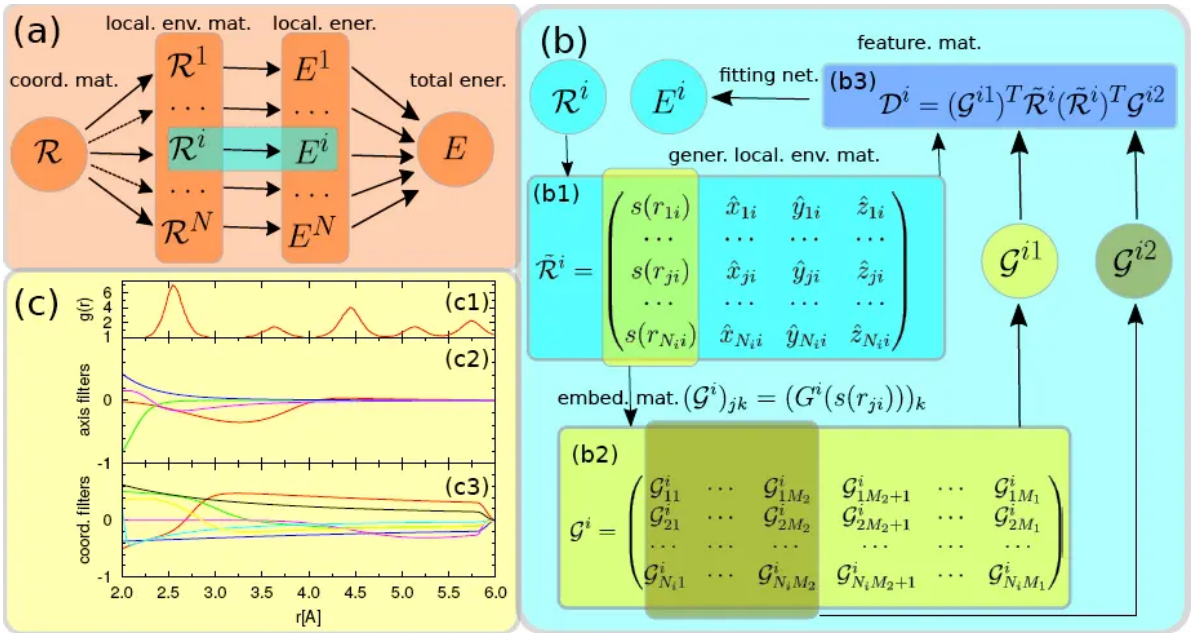
$$\mathcal{R}^i = \{r_{1i}^T, \cdots, r_{ji}^T, \cdots, r_{N_i,i}^T\}^T,\ r_{ji} = (x_{ji}, y_{ji}, z_{ji})$$

$$\{x_{ji}, y_{ji}, z_{ji}\} \mapsto \{s(r_{ji}), \hat{x}_{ji}, \hat{y}_{ji}, \hat{z}_{ji}\}$$

$$\hat{x}_{ji} = \frac{s(r_{ji})x_{ji}}{r_{ji}},\ \hat{y}_{ji} = \frac{s(r_{ji})y_{ji}}{r_{ji}},\ \hat{z}_{ji} = \frac{s(r_{ji})z_{ji}}{r_{ji}},\ \text{and } s(r_{ji})$$

$$s(r_{ji}) = \begin{cases} \dfrac{1}{r_{ji}}, & r_{ji} < r_{cs}. \\ \dfrac{1}{r_{ji}}\left\{\dfrac{1}{2}\cos\left[\pi\dfrac{(r_{ji}-r_{cs})}{(r_c-r_{cs})}\right]+\dfrac{1}{2}\right\}, & r_{cs} < r_{ji} < r_c. \\ 0, & r_{ji} > r_c. \end{cases}$$

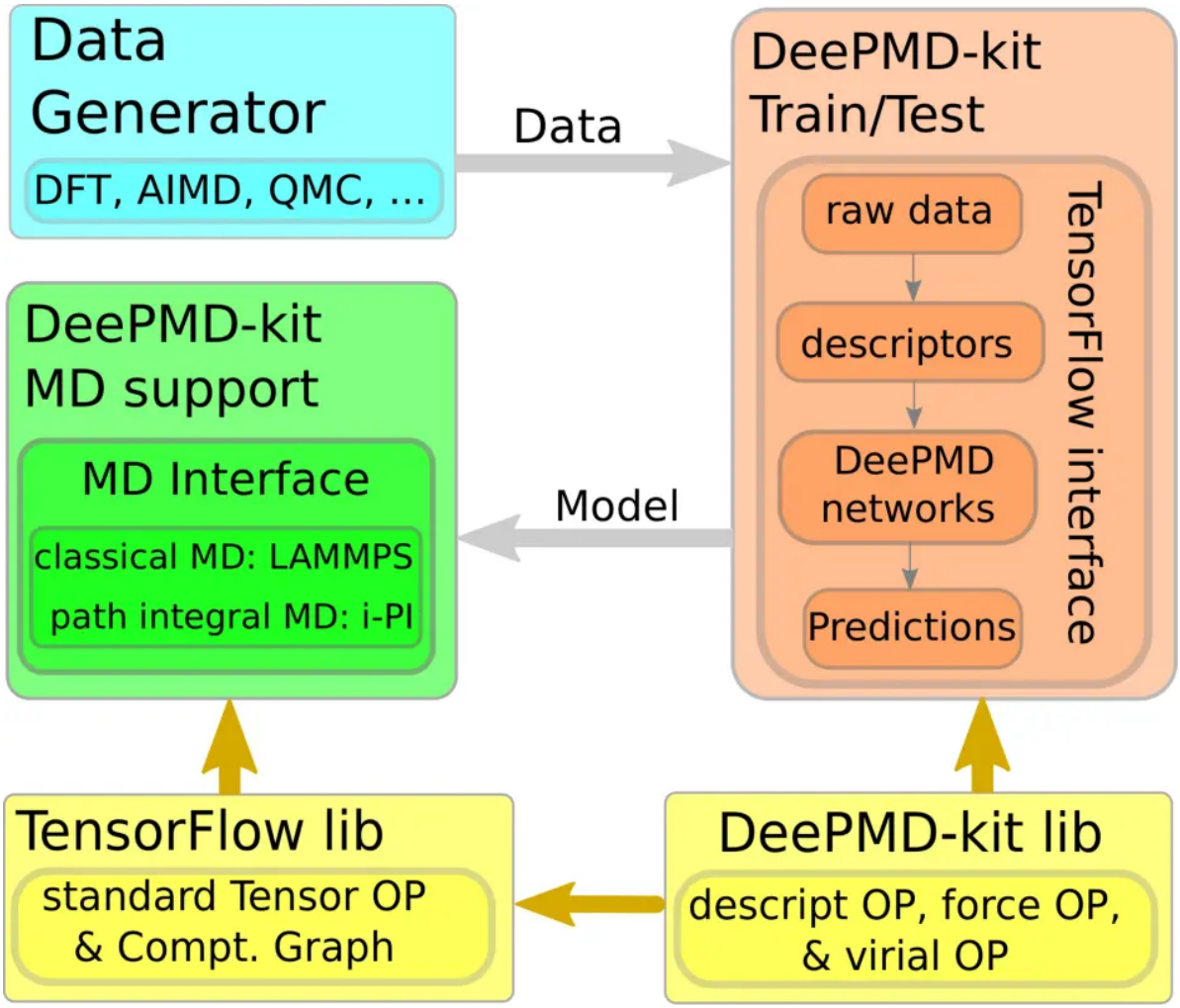
为保证旋转和置换的对称性，我们引进内嵌矩阵：



DeePMD框架

DeePMD-kit包括三部分：1) 用于计算作用力、描述符和作用力的C++库，包括与TensorFlow的接口和第三方MD包；2) 使用TensorFlow的训练和测试程序；3) 对于LAMMPS和i-PI的支持。

一个完整的模拟过程包括以下步骤：1) 对于给定的系统，DeePMD-kit先将AIMD计算得到的数据转化为一种自定义的文件格式，其中包括原子的坐标以及原子的能量、力和virial。2) 随后输入到由TensorFLow框架搭建的势函数网络进行训练，通过原子坐标预测此时的能量。3) 训练好的模型将被冻结保存，提供给融合了DeePMD-kit的传统分子动力学模拟软件如LAMMPS。4) 使用LAMMPS对给定的数据进行分子动力学模拟。



DeePMD安装和使用

DeePMD安装

下载anaconda

去官网下载Linux版本并sh安装。

使用conda直接安装DeePMD-kit

安装conda后，使用以下命令安装GPU版：

```
conda install deepmd-kit=*=*gpu lammps-dp=*=*gpu -c deepmodeling
```

将gpu改为cpu即可安装CPU版：

```
conda install deepmd-kit=*=*cpu lammps-dp=*=*cpu -c deepmodeling
```

如需指定版本，需将两个等号中间的*号改为版本号：

```
conda install deepmd-kit=1.3=*cpu lammps-dp=1.3=*cpu -c deepmodeling
```

离线安装

也可以至https://github.com/deepmodeling/deepmd-kit/releases下载离线安装包，或用wget：

```
wget https://github.com/deepmodeling/deepmd-kit/releases/download/v1.3.1/deepmd-kit-1.3.1-cuda10.1_gpu-Linux-x86_64.sh -O deepmd-kit-1.3.1-cuda10.1_gpu-Linux-x86_64.sh
```

以1.3.1 GPU版为例，下载后执行以下命令，按提示操作即可。

```
sh deepmd-kit-1.3.1-cuda10.1_gpu-Linux-x86_64.sh
```

docker安装

拉取GPU版本：

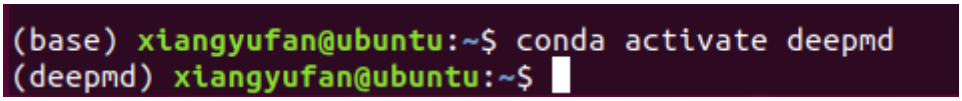
```
docker pull ghcr.io/deepmodeling/deepmd-kit:1.3.1_cuda10.1_gpu
```

拉取CPU版本：

```
docker pull ghcr.io/deepmodeling/deepmd-kit:1.3.1_cpu
```

DeePMD使用

激活DeePMD-kit环境，输入conda activate DeePMD即可从基础环境切换为DeePMD的环境。



可用指令：

```
Valid subcommands:
  {config,transfer,train,freeze,test,compress,doc-train-input,model-devi,convert-from}
    config              fast configuration of parameter file for smooth model
    transfer            pass parameters to another model
    train               train a model
    freeze              freeze the model
    test                test the model
    compress            compress a model
    doc-train-input     print the documentation (in rst format) of input
                        training parameters.
    model-devi          calculate model deviation
    convert-from        convert lower model version to supported version
```

train

官网上的示例：

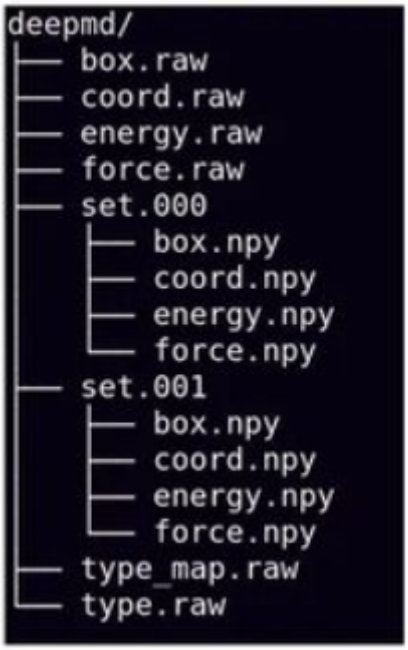
Several examples of training can be found at the `examples` directory:

```
$ cd $deepmd_source_dir/examples/water/se_e2_a/
```

After switching to that directory, the training can be invoked by

```
$ dp train input.json
```

当然如果要跑自己的数据集，一是要转换训练集格式(更多请阅读dpdata文档)。



二是要配置input.json达到自己的需求。上述示例中input.json如下，我个人加了一些注释作为参考。

```
{
    "_comment": " model parameters",
    "model": {
    "type_map": ["O", "H"],//势函数包含元素种类，水H2O，有H和O两种元素
    "descriptor" :{
        "type":     "se_e2_a",//类别
        "sel":      [46, 92],//本例为截断半径里最多容纳46个氢原子和92个氧原子
        "rcut_smth":    0.50,//s(rij)公式中开始平滑的半径rc
        "rcut":     6.00,//s(rij)公式中截断半径rc
        "neuron":       [25, 50, 100],//神经网络大小25*50*100
        "resnet_dt":    false,//传参网络开关
        "axis_neuron":  16,//内嵌矩阵embed.mat(G1)jk的宽度
        "seed":     1,//随机数生成器种子
        "_comment":     " that's all"
    },
    "fitting_net" : {
        "neuron":       [240, 240, 240],//神经网络大小
        "resnet_dt":    true,////传参网络开关
        "seed":     1,//随机数生成器种子
        "_comment":     " that's all"
    },
    "_comment": " that's all"
    },

    "learning_rate" :{
    "type":     "exp",//衰减方式
    "decay_steps":  5000,//每5000步衰减一次学习率
    "start_lr": 0.001,//初始学习率
    "stop_lr":  3.51e-8,//最终学习率
    "_comment": "that's all"
    },

    "loss" :{
    "type":     "ener",
    "start_pref_e": 0.02,//初始能量误差在loss函数中的权重
    "limit_pref_e": 1,//最终能量误差在loss函数中的权重
```

```
    "start_pref_f": 1000,//初始力在loss函数中的权重
    "limit_pref_f": 1,//最终力在loss函数中的权重
    "start_pref_v": 0,//初始维里应力(张量？)在loss函数中的权重
    "limit_pref_v": 0,//最终维里应力(张量？)在loss函数中的权重
    "_comment": " that's all"
    },

    "training" : {
    "training_data": {
        "systems":      ["../data/data_0/", "../data/data_1/",
"../data/data_2/"],//指定训练所在文件夹的位置
        "batch_size":   "auto",//同时训练数量
        "_comment":     "that's all"
    },
    "validation_data":{
        "systems":      ["../data/data_3"],//指定训练所在文件夹的位置
        "batch_size":   1,//同时训练数量
        "numb_btch":    3,//模型验证的批数
        "_comment":     "that's all"
    },
    "numb_steps":   1000000,//训练步数
    "seed":     10,//随机数种子
    "disp_file":    "lcurve.out",//打印学习曲线的文件
    "disp_freq":    100,//打印学习曲线的频率，以步为单位
    "save_freq":    1000,//保存检查点的频率
    "_comment": "that's all"
    },

    "_comment":     "that's all"
}
```

freeze

指令如下：

```
dp freeze                       Model file:frozen_model.pb
dp freeze -o graph.pb           Model file:graph.pb
```

test

指令如下：

```
dp test -m graph.pb -s /root/workshop/deepmd-kit/data/test/ -d result
dp test -m graph.pb -s /root/workshop/deepmd-kit/data/ -d result
```

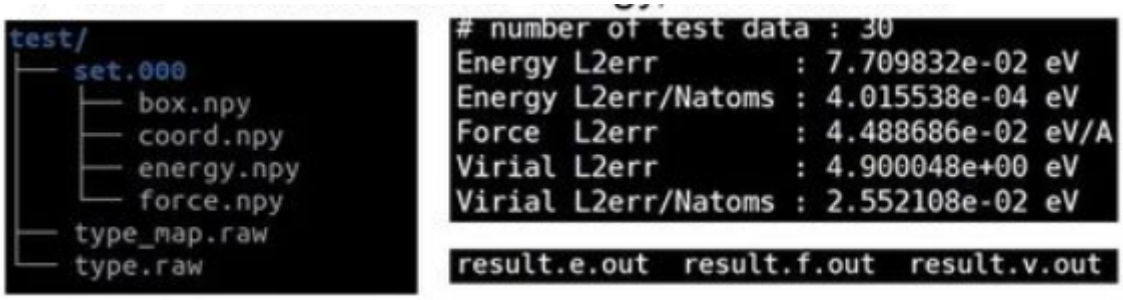
其中

-m Model file graph.pb

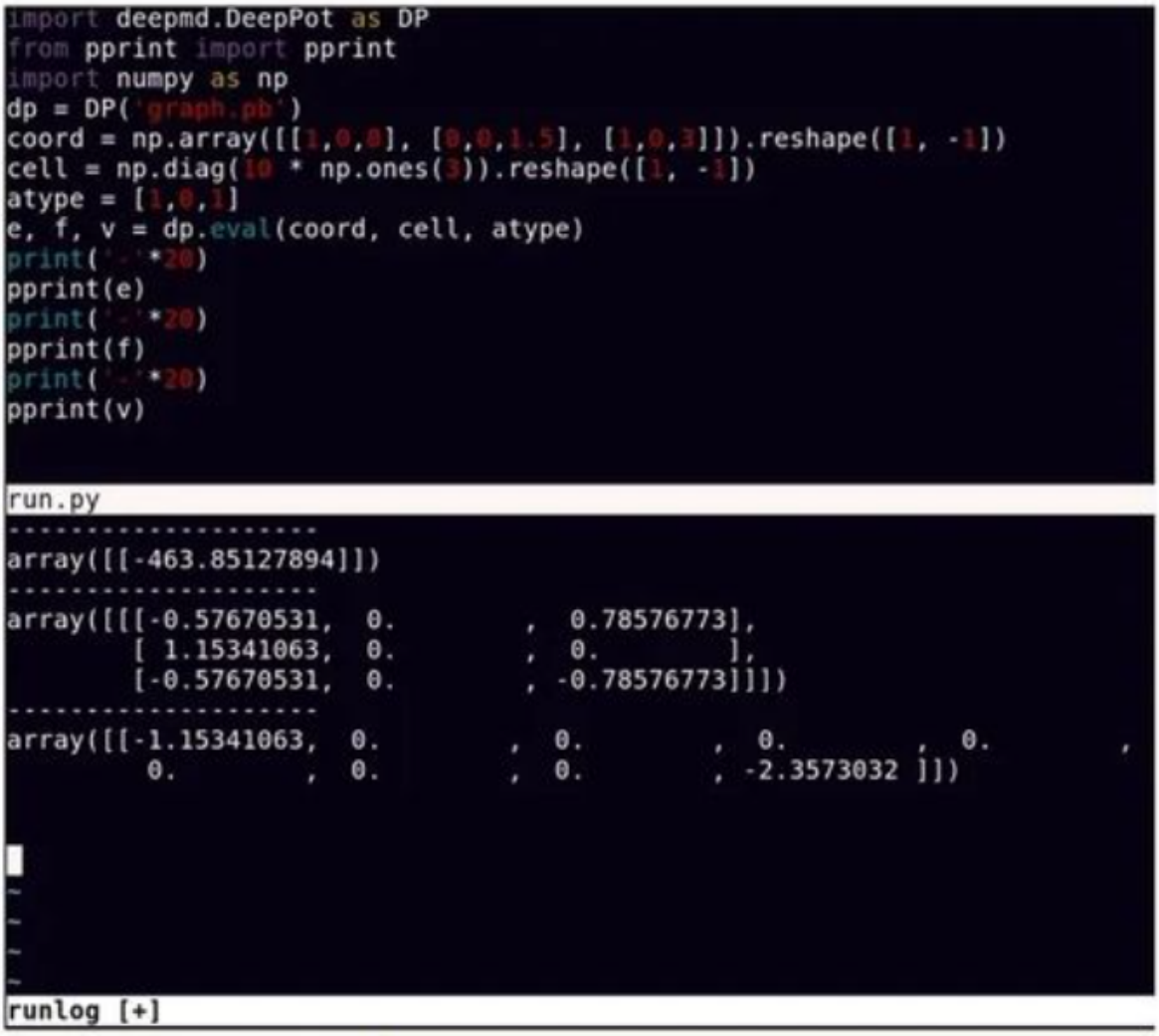
-s test set directory

-d save detail info. about energy, force and viral

模型的质量由test results和lcurve.out来评估。



python接口



参考资料

[1]H. Wang, L. Zhang, J. Han, and W. E, “DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics,” Computer Physics Communications, vol. 228, pp. 178–184, 2018.

[2]Zhang, Linfeng , et al. "End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems." (2018).

[3] https://github.com/deepmodeling/deepmd-kit

[4]https://www.zhihu.com/zvideo/1351912744078315520（DeePMD-kit官方教程）

[5]https://zhuanlan.zhihu.com/p/347578797

[6]https://zhuanlan.zhihu.com/p/375508158