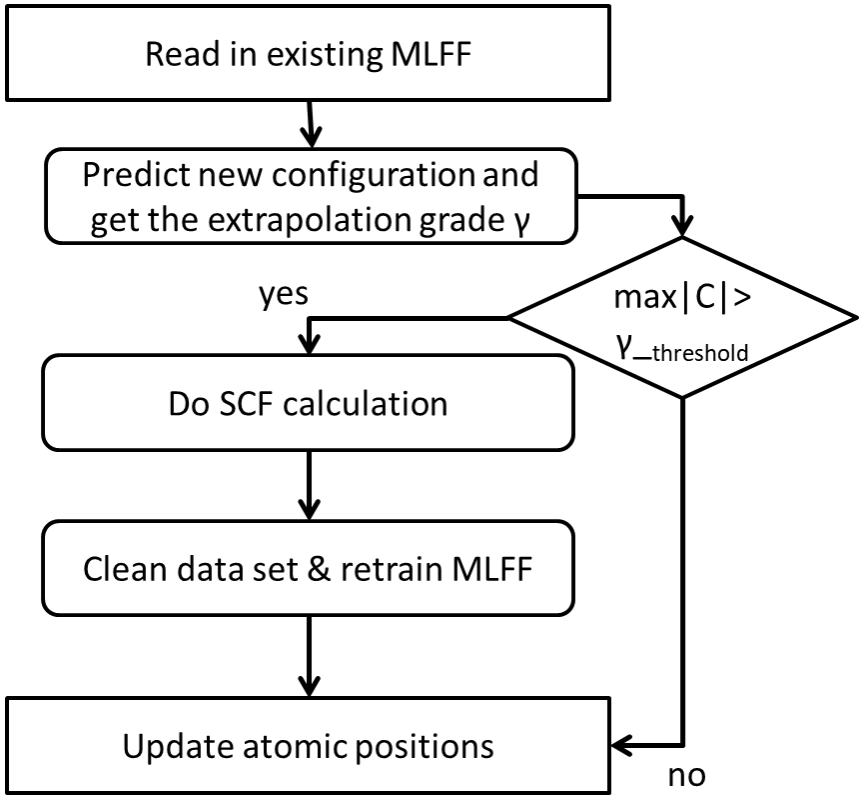
On the fly 做了什么？

MLFF on the fly 的基本思想是，在利用 MLFF 进行 MD 模拟的每一个步骤中，都对这个结构进行一次判断，并且求得一个参数 γ（extrapolation grade），它的意义就是度量了这个结构是否能够被现有训练集良好的预测。当 γ 大于设定的阈值时，则会进行 DFT 计算并将 DFT 得到的力和能量加入训练集并替换掉之前被判断表现的不好的数据，并对新的训练集重新训练出新的模型用于接下来的 MLFF 的预测。流程图如下



在这里我们用 Moment Tensor Potentials（MTPs）来作为 ML interatomic interaction 的模型。在这个模型里，简单来说就是把总的能量分成来自每个原子上的贡献

$$\mathrm{V}(\mathrm{r_i}^{(\mathrm{k})}) \equiv \sum_{j=1}^{m} \theta_j B_j(r_i^{(k)}) \quad , \quad 1 \le k \le K, 1 \le i \le N^{(k)}$$

上式 B 就是 MLFF 训练得到的由原子位置信息转变而来的特征，θ 为拟合的参数。
在上文中提到的并且是这个方法的关键便是对 extrapolation grade γ的定义[1]。在这里我们将 γ 定义为矩阵 C 的最大的那个矩阵元，其中矩阵 C 定义为

$$C = \begin{pmatrix} B_1(r^*_1) & \cdots & B_m(r^*_1) \\ \vdots & \ddots & \vdots \\ B_1(r^*_N) & \cdots & B_m(r^*_N) \end{pmatrix} \mathrm{A}^{-1}$$

在解释公式出现的 A 矩阵的定义之前，需要先得到另一个长矩阵 A_all。在 A_all 中是我们由

MLFF 模型训练得到的特征，我们将其收集起来得到了一个很长的矩阵。

$$A\_all = \begin{pmatrix} B_1(r_1^{(1)}) & \cdots & B_m(r_1^{(1)}) \\ \vdots & \ddots & \vdots \\ B_1(r_N^{(K)}) & \cdots & B_m(r_N^{(K)}) \end{pmatrix}$$

而之前提到的 A 矩阵就是从这个长矩阵 A_all 中得到的一个拥有最大的行列式的方形的子矩阵[2]，可以用于求取上述矩阵 C。

其中对 A_all 取子矩阵的方法叫做 maxvol algorithm，这里简单介绍一下它的算法

1. 先对矩阵 A_all 随机的取一个 m*m 的方阵 A，剩下备用的为 B，计算 $C=B*A^{-1}$
2. 寻找矩阵 C 中绝对值最大的一个值 C_{ij}
3. 当$|C_{ij}|$>设定的阈值，则：
   - 3.1 交换 A 的第 i 行与 B 的第 j 行
   - 3.2 更新 $C=B*A^{-1}$，这里可以用 Sherman-Morrison 公式来求逆
   - 3.3 回到第 2 步

最后就可以得到最大行列式的 A 了，设定的阈值越小，A 的行列式越大。想要直观的去理解这个过程，我们可以认为不断替换矩阵元的过程实际上就是在从这一个长矩阵中寻找一个最线性无关的子矩阵。那么得到的这个子矩阵就更能代表整个训练集。

上述方法可以参考文献

[1]http://dx.doi.org/10.1016/j.commatsci.2017.08.031
[2]https://doi.org/10.1142/9789812836021_0015

需要准备什么？

1. 安装问题

1.1 编译 MLFF_0523，从网站上下载得到 on_the_fly_test.zip 后解压得到 MLFF_0523.tgz，再次解压后进入 MLFF_0523/src 通过$ sh build.sh 编译，成功后检查环境内是否可以调用 main_MD.x

1.2 安装 ase（有 pwmat 接口的版本），从网站上下载得到的 on_the_fly_test.zip 内解压得到 ase.tgz。（1）通过$conda create -n MLFF_0523 python=3.8 建议一个新环境，并通过 $conda activate MLFF_0523 激活环境；（2）通过$pip install ase 安装一个官方版本的 ase，在命令行中使用$python 在其中通过 import ase ; ase.__file__来查看其根目录所在位置

```
[ycpan@mstation ~]$ python
Python 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 17:14:51)
[GCC 7.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import ase
>>> ase.__file__
'/home/ycpan/.local/lib/python3.6/site-packages/ase/__init__.py'
```

（3）将 ase.tgz 解压得到的文件夹 ase 替换掉刚才得到地址的 /home/ycpan/.local/lib/python3.6/site-packages/ase 文件

1.3 安装 python 库 maxvolpy，通过$ pip install maxvolpy 来安装，安装完后注意检查能否成功调用

2. 在需要运行的初始结构（需命名为 md.atom.config）下解压运行所需的文件，并在当前目录下建立文件夹 PWdata/data1/，并且将之前准备好的 AIMD 得到的 MOVEMENT（在计算 AIMD 时就需要在它的 etot.input 内**设置 energy_decomp**，否则跑不了！！）移动到 PWdata/data1/下

3. 参考 mlff 手册（http://modulefiles.pwmat.com/pwmat-resource/module-download7/pdf/mlff_manual.pdf）准备 parameters.py 和 md.input，并将压缩包 on_the_fly_test.zip 内的 on_the_fly.tgz 解压缩至当前文件夹下

4. 修改 main_qs3.py

需要修改的文件 main_qs3.py，打开该文件，在 Import 之后有一系列需要自行修改的参数输入

nnodes 跑 main_MD.x 需要的节点数

nsteps 只能是 1

md_steps_all 程序运行的最大次数

end_steps 实际上要运行的 MD 步数

init_movement_nsteps 初始训练集有多少（PWdata/data1/MOVEMENT 内的 image 数）

md_atom_config 初始结构文件名

command_ase 运行 ase 使用的命令，值得注意的是 mlff 环境与 PWmat 下的 mpirun 并不一样，所以需要用绝对路径来调用 mpirun，例如：

```
command_ase='/opt/openmpi/2.1.0/intel-cuda/bin/mpirun -np 4 PWmat'
```

th_1 : extrapolation grade 的下限，判断每一个结构是否是可以被当前训练集合理预测的参数，默认为最小值 1，大于 1 的话可以减少计算量但是会逐渐降低模型的精度。（按我个人使用的经验来看的话，1.001、1.01 也可以有效降低计算量而且模型精度也不会太大幅度的变化。不是非常确定，需要自行尝试）
th_2 : extrapolation grade 的上限，如果超过这个值的话说明训练集和当前需要预测的结构差距太大，会退出程序。可以设置成 100 或 1000 之类的保证程序不会停止，但如果太大了，还是建议先多跑点 AIMD 扩充一下训练集。
ts_natom 训练集内结构的原子数
ntypes 使用的特征种类。类型是列表，需要写在列表内如[1,2]
flag1 开始的第一步是不是续算，True/False
in_mdopt 需不需要 IN.MDOPT，True/False
is_tempchange 需要升温或者降温的时候设置，True/False
temp_range 温度列表，在 is_tempchange 为 True 时才会读取，用来设置末态温度的变化。类型是列表，如[600,2000]
check_interv 隔多少步做一次 scf 用于比较 mlff 预测的结果与 dft 结果的差别，会输出这一步的 force 和 atomic energy 的 RMSE

除此之外还需要更改的是 main_qs3.py 中第 108 行 calc 的参数，把 etot.input 的参数写入其中，值得注意的是需要与得到训练集时候的参数保持一致

```
calc = PWmat(kpts=(1,1,1,0,0,0,2),IN_PSP1='Li.SG15.PBE.UPF',IN_PSP2='Ge.SG15.PBE.UPF',IN_PSP3='P.SG15.PBE.UPF',IN_PSP4='S.SG15.PBE.UPF'
      ,ENERGY_DECOMP='T',Ecut=60,node1=4,node2=1,command=command_ase)
```

当一切准备完毕后，通过$ python main_qs3.py 即可运行

使用时需注意

在运行过程中这个 code 会不断修改你的训练集，md.input 和 parameters.py，首先你需要备份初始的训练集，即 AIMD 得到的 MOVEMENT，以及输入文件。如果当你中断程序的运行后，需**更改你的 md.input 和 parameters.py** 才能再次运行。