The MLIP package User Manual

Supplemental Information for the Manuscript
"The MLIP package: Moment Tensor Potentials with MPI and Active Learning"

I. Novikov, K. Gubaev, E. Podryabinkin, A. Shapeev

July 15, 2020

1 Installation of the MLIP code and license

MLIP is a software package implementing moment tensor potentials. It is distributed for free for non-commercial purposes at https://mlip.skoltech.ru/download/. Here we describe how to install and use the MLIP package in a Linux-like system.

After a user is registered and added to the repository, the user can obtain the second version of the code (described in [1]) through git by executing

```
git clone https://gitlab.com/ashapeev/mlip-2.git
```

The command will create the `mlip-2/` folder with the source files. For the MLIP package installation execute the following two commands (in a UNIX-like terminal):

```
./configure
make mlp
```

These commands will generate the binary file `mlp` in the `bin/` folder. This is the executable file which allows to run the MLIP commands including: conversion of files from/to the internal `.cfg` to/from files in other formats; training of MTP; calculation of grades for configurations from a scratch; selection/adding configurations from a scratch to a training set, etc.

The most frequently used MLIP commands are described in Section 5.

2 CFG file

`.cfg` files are text-format files containing datasets of configurations, with or without computed energies, forces, and stresses. Each `.cfg` can contain zero or more configurations; an example of a `.cfg` file is:

```
BEGIN_CFG
 Size
  4
 Supercell
  2.86      0.00      0.00
  0.00      5.71      0.00
  0.00      0.00      4.05
 AtomData:  id type cartes_x cartes_y cartes_z     fx     fy     fz
             1    0     0.00    -0.13     0.00   0.00   0.75   0.00
             2    0     1.43     1.45     2.06   0.00  -0.13   0.00
             3    1     0.00     2.87     0.00   0.00  -0.41   0.00
             4    1     1.43     4.26     2.06   0.00  -0.20   0.00
 Energy
  -16.023765555539
 PlusStress:  xx     yy     zz     yz     xz     xy
           -0.29   0.23  -0.09   0.00   0.00   0.00
Feature   EFS_by   vasp
Feature   from     database:p.cfg
Feature   mindist  2.70
END_CFG

BEGIN_CFG
 Size
  1
 Supercell
   2    0   0
   1  1.7   0
 AtomData:  cartes_x cartes_y cartes_z
            0 0 0
END_CFG
```

The description of each configuration in the database starts with `BEGIN_CFG` and ends with `END_CFG`. No symbols except whitespace (which is space, tab, and the newline characters '\n' and '\r') are allowed between configurations. After `BEGIN_CFG` several compulsory and optional fields follow. Bellow is the list of recognizable fields.

1. `Size` (compulsory field) is followed by one integer number, the number of atoms in the configuration. `Size` should appear before "`AtomData:`".

2. `Supercell` (optional field) is followed by 3, 6, or 9 numbers, the 3 coordinates of the first lattice vector, then (optionally) 3 coordinates of the second lattice vector, and (optionally) 3 coordinates of the third lattice vector. If `Supercell` is missing then no lattice vectors are given (this is a open-shell molecule).

3. "`AtomData:`" (compulsory field) contains the per-atom data (atomic coordinates, forces, atom types, etc.). `AtomData:` is followed by one or more field names (on the same line):

   (a) `id` (optional field) means the ordinal number of atom in the configuration (1, 2, ...). If `id`'s are given, but are different from 1, 2, 3 in exactly in this order, it is an error.

(b) `type` (optional field) are the element types (0, 1, ...); if they are empty then all atoms are considered to be of type 0 and this is a single-component configuration.

(c) `cartes_x`, `cartes_y`, `cartes_z` (all three are compulsory fields) means the Cartesian coordinates (measured in Å). There is an alternative option to provide the configuration with direct (fractional) coordinates. In this case `direct_...` caption should be specified instead of `cartes_...`.

(d) `fx`, `fy`, `fz` (optional fields, but all three are required if at least one is present), are the force components in Cartesian coordinates (measured in eV/Å).

4. `Energy` (optional field) is followed by one number (measured in eV).

5. "`PlusStress:`" (optional field) must be immediately (on the same line) followed by the following 6 (not more and not less) field names (in any order), `xx`, `yy`, `zz`, `yz`, `xz`, and `xy`. On the next line are the 6 numbers, corresponding to those stresses. These are virial stresses multiplied by the cell volume (measured in eV). Positive stresses correspond to compressed configurations.

6. "`Feature`" (optional fields, multiple entries are allowed) is/are additional (textual) attributes of a configuration (such as what chemical elements corresponds to the atom indexes, what generated it, what computed its energy, whether it is an ideal crystal or has a defect, etc.). Each "Feature" has a name (one word without whitespace) and value (a string separated from the name by a space or a tab character). A configuration may contain several features. If two lines with the same name appear, the newline ('`\n`') character and the second line are appended to the first line; similarly for the third line, etc. The features do not have universal meanings, they are used by different tools.

3 MTP file

`.mtp` files encode a functional form of MTP allowing for calculation energy, forces, and stresses of a configuration. Here we give an example of the `.mtp` file describing the MTP of the level 8 before and after training. The example of an untrained MTP is shown below.

```
MTP
version = 1.1.0
potential_name = MTP1m
species_count = 1
potential_tag =
radial_basis_type = RBChebyshev
    min_dist = 2.25
    max_dist = 6.2
    radial_basis_size = 8
    radial_funcs_count = 2
alpha_moments_count = 18
alpha_index_basic_count = 11
alpha_index_basic = {{0, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0},
```

```
{0, 0, 0, 1}, {0, 2, 0, 0}, {0, 1, 1, 0}, {0, 1, 0, 1},
{0, 0, 2, 0}, {0, 0, 1, 1}, {0, 0, 0, 2}, {1, 0, 0, 0}}
alpha_index_times_count = 14
alpha_index_times = {{0, 0, 1, 11}, {1, 1, 1, 12}, {2, 2, 1, 12},
{3, 3, 1, 12}, {4, 4, 1, 13}, {5, 5, 2, 13}, {6, 6, 2, 13},
{7, 7, 1, 13}, {8, 8, 2, 13}, {9, 9, 1, 13}, {0, 10, 1, 14},
{0, 11, 1, 15}, {0, 12, 1, 16}, {0, 15, 1, 17}}
alpha_scalar_moments = 9
alpha_moment_mapping = {0, 10, 11, 12, 13, 14, 15, 16, 17}
```

There are two groups of fields in the file: those that could be changed by a user, and unchangeable ones related to the level and a particular functional form of the MTP. We first describe the hyperparameters that could be adjusted by the user.

`species_count` is the number of components (atomic types) in the system investigated;

`radial_basis_type` is the type of radial basis used (the most frequently used radial basis is the Chebyshev basis);

`min_dist` is the minimal distance between atoms in the training set (in Angstroms);

`max_dist` is the cutoff radius (in Angstroms);

`radial_basis_size` is the size of radial basis (the Chebyshev basis for the given MTP).

The second group of fields cannot be freely modified by a user. They are:

`radial_funcs_count`
: is the number of radial functions f_μ *[IN: I think it better not to write any notations here used in the paper]* which form the moment tensor descriptors of the MTP of the given level (e.g., for level 8 it is necessary to consider two radial functions);

`alpha_moments_count`
: is the total number of the scalar components of all the (scalar-valued and not) basis functions of the MTP of a given level;

`alpha_index_basic_count`
: is the total number of the scalar components of all the moment tensor descriptors of the MTP of a given level;

`alpha_index_basic`
: define those scalar components. The entries of this field include the indices of radial functions (the first index, could be equal to 0 or 1, because only two radial functions needed to construct all MTP basis functions), and the powers of the components of the vector joining the atomic neighbors, x_{ij}, y_{ij}, z_{ij} (the second, the third, and the fourth indices, correspondingly) *[IN: Again, I think we do not need any notations here]*;

`alpha_index_times_count`
: is the number of scalar multiplications required to compute all the basis functions;

`alpha_index_times`
: define those multiplications. The entries of this field include the indices of two scalars being multiplied (the first and the second indices), the scalar prefactor (the third index), and the index of the resulting scalar (the fourth index);

`alpha_scalar_moments`
: is the number of the scalar-valued MTP basis functions; and, finally,

`alpha_moment_mapping`
: are the indices of those basis functions.

For example, for the current MTP of the level 8 there are 9 MTP basis functions

$$\begin{array}{ll} B_1 = M_{0,0}, & \mathrm{lev} B_1 = 2; \\ B_2 = M_{1,0}, & \mathrm{lev} B_2 = 6; \\ B_3 = M_{0,0}^2, & \mathrm{lev} B_3 = 4; \\ B_4 = M_{0,1} \cdot M_{0,1}, & \mathrm{lev} B_4 = 6; \\ B_5 = M_{0,2} : M_{0,2}, & \mathrm{lev} B_5 = 8; \\ B_6 = M_{0,0} M_{1,0}, & \mathrm{lev} B_6 = 8; \\ B_7 = M_{0,0}^3, & \mathrm{lev} B_7 = 6; \\ B_8 = M_{0,0}(M_{0,1} \cdot M_{0,1}), & \mathrm{lev} B_8 = 8; \\ B_9 = M_{0,0}^4, & \mathrm{lev} B_9 = 8; \end{array} \tag{1}$$

therefore, `alpha_scalar_moments` = 9 and we need the objects with the indices `alpha_moment_mapping` to form the MTP basis functions (1) *[IN: Probably, remove (1), we have it in the paper]*.

After the fitting of the MTP, the parameters of fitting added in `.mtp` file: the vectors with `radial_basis_size` × `radial_funcs_count` radial coefficients

```
radial_coeffs
    0-0
        {-4.243215058993401e-01, -7.013766150563229e-01, ... }
        {8.483313776829586e-02, -2.944665241055562e-02, ... }
```

the vector with energies of individual atoms, one per atomic type,

```
species_coeffs = {-4.699403701006758e+00}
```

and the vector with 9 linear coefficients

```
moment_coeffs = {1.0466030834e+00, 6.5878384919e-01, ...}
```

4 ALS file

A `.als`-file stores the active learning state which includes the MTP, the active set, and the corresponding matrices. It has a mixed text/binary structure and is organized as follows. First goes the fitted MTP. Next the selection weights are specified. After that—the binary part follows. It contain a binary representation of active selection matrices A and A^{-1}. Actually these matrices can be recalculated from the active set that is stored below, however this data helps to prevent the change of the active learning state after loading due to rounding errors. Next, a block with the configurations corresponding to the matrix rows are stored. At the end of the `.als`-file the active set (in the `.cfg`-format with full floating-point precision) is writen.

5 mlp binary file

The `mlp` binary file allows the user to run various MLIP code commands. To list the most frequently used MLIP commands, execute `mlp list`. To provide a description of a certain command, execute `mlp help [command]`. A typical template of the rest commands available is

```
mlp [command] [input/output files] [options]
```

The main MLIP commands and description are given below.

`mlp relax mlip.ini --cfg-filename=to_relax.cfg --save-relaxed=relaxed.cfg`
Relaxation of configurations read from the file `to_relax.cfg` in accordance with the settings in `mlip.ini`, writing the result to the file `relaxed.cfg`.

`mlp convert-cfg <input-filename> <output-filename> --options`
Reading configurations from `input-filename` and writing them to `output-filename`, possibly converting to a different format. All the formats, except for internal MLIP `.cfg`, should be specified explicitly, e.g.

```
mlp convert-cfg input.cfg POSCAR --output-format=vasp-poscar
```

(in other words, `mlp` does not try to guess the format based on the contents or the filenames).

`mlp train init.mtp train.cfg --trained-pot-name=pot.mtp`
Training of MTP as given by `init.mtp` file on the dataset `train.cfg`. The option `trained-pot-name` indicates that the potential trained should be written to `pot.mtp`.

`mlp mindist train.cfg`
Calculation of the minimal distance between atoms for each of the configurations in `train.cfg`.

`mlp calc-errors pot.mtp db.cfg`
Calculation of the prediction errors by `pot.mtp` on the database `db.cfg`.

`mlp calc-efs pot.mtp in.cfg out.cfg`
Calculation of energies, forces, and stresses for the configurations in `in.cfg` with `pot.mtp`, writing the result to `out.cfg`.

`mlp calc-grade pot.mtp train.cfg in.cfg out.cfg --als-filename=state.als`
First, creation of the `state.als` file with the active learning state for the potential `pot.mtp` fitted on the database `train.cfg`. Then, calculation of the extrapolation grades of `in.cfg`, writing the configurations with the grades to `out.cfg`.

`mlp select-add pot.mtp train.cfg preselected.cfg add_to_train.cfg`
Maxvol selection of non-repetitive configurations from `preselected.cfg`, writing the selected configurations in `add_to_train.cfg`. After calculating the selected configurations on an ab initio model they are supposed to be added to the training set.

6 MLIP settings file

A settings file, typically named as `mlip.ini`, is required when MLIP is used from LAMMPS as `pair_style`, namely

```
pair_style  mlip mlip.ini
pair_coeff  * *
```

The same file can be used with the `mlp relax` command to specify what MTP file to use and whether or how active learning should be used. It contains the instructions and settings for MLIP determining the operational regime. An example of a simple `mlip.ini` is below:

```
mtp-filename          pot.mtp
select                FALSE   # turned off for a large-scale MD run
```

More examples of settings file can be found in examples provided with the MLIP package.

The `mlip.ini` file is organized according the following rules.

- Each setting is given in a separate line and have the form

  ```
  identifier    value
  ```

  where `identifier` is the string without whitespace and `value` is another string without whitespace (a string can be a number). They must be separated by any number of spaces or tab symbols. Whitespace before `identifier` as well as any symbols after `value` are allowed but ignored by the parser.

- Other than identifiers and values, the lines of the settings file can contain comments. Comments begin with "`#`" and all the foregoing symbols are ignored in this line.

- The order of lines is not important; therefore the lines can be grouped for the user's convenience.

- If a parameter is not present (or commented out), its default value is used.

- If a setting is not supported then it is ignored (without a warning).

The recognizable identifiers and their meaning are listed below.

`mtp-filename <filename>`
: File name with MTP to be loaded.

`write-cfgs <filename>`
: if this setting is present then the processed configurations will be written to the specified file, may be useful for debugging.

`write-cfgs:skip <number>`
: skip this many configurations before writing a processed configuration. Default value is 0.

`select <TRUE/FALSE>`
: activates or deactivates calculation of extrapolation grades and optionally writing configurations with high extrapolation grades. If the `FALSE` value is specified the following settings will be ignored.

`select:threshold <real number>`
: corresponds to γ_{select} mentioned in [1][Section 2.3]. If the extrapolation grade exceeds this value then the configuration will be saved to the specified file.

`select:threshold-break <real number>`
: corresponds to γ_{break} mentioned in [1][Section 2.3]. If the extrapolation grade exceeds this value the program execution will be interrupted, that can be used for prevention of instability of the simulation.

`select:save-selected <filename>`
: the file for saving the configurations whose grade exceeds `select:threshold`.

`select:load-state <filename>`
: the file for loading the active learning state, typically created by the `mlp calc-grade` command.

`select:log <filename>/stdout/stderr`
: a file (or standard output stream) for writing a log of the configuration selection process.

References

[1] I. Novikov, K. Gubaev, E. Podryabinkin, and A. Shapeev. The MLIP package: Moment Tensor Potentials with MPI and active learning. Manuscript.