TorchANI Worknotes

Fadjar Fathurrahman

1 Installation

Prerequisites: PyTorch

Recommendation: install to user directory.

The usual steps are applicable:

- `python setup.py build`
- `python setup.py install --user`

TorchANI will be installed at (for example, the version number may vary):

```
$HOME/.local/lib/python3.8/site-packages/torchani-2.1.2-py3.8.egg/
```

To check the installation, go to directory examples and run the `energy_force.py` example. TorchANI will download the necessary files (paramaters for the neural network) so the script might run slowly in the first call. The parameters will be put under the `resources` subdirectory of TorchANI.

There is warning when using Torch v-1.6:

```
/home/efefer/.local/lib/python3.8/site-packages/torchani-2.1.2.dev12+g42442af-py3.8.egg/torchani/aev.py:
↪  UserWarning: This overload of nonzero is deprecated:
        nonzero()
Consider using one of the following signatures instead:
        nonzero(*, bool as_tuple) (Triggered internally at
        ↪  /opt/conda/conda-bld/pytorch_1595629395347/work/torch/csrc/utils/python_arg_parser.cpp:766.)
in_cutoff = (distances <= cutoff).nonzero()
```

The fix seems to be:

```
in_cutoff = torch.nonzero(distances <= cutoff, as_tuple=False)
```

2 CPU vs GPU comparison

Script `test_cnt_5x5.py`. Timing using time command in Linux.

```
from ase.build import nanotube
from ase.md.langevin import Langevin
from ase.optimize import BFGS
from ase import units

import torch
import my_torchani

# Now let's set up a crystal
atoms = nanotube(5, 5, length=10, bond=1.420, symbol='C')
print(len(atoms), "atoms in the cell")
atoms.set_pbc([True,True,True])
Lz = atoms.cell.array[2,2]
Lx = 20.0
Ly = 20.0
atoms.set_cell([Lx, Ly, Lz])
atoms.center()
atoms.write("STRUCT_cnt_5x5.xsf")

#device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device = "cpu" # set manually to cpu
print("device = ", device)

# Now let's create a calculator from builtin models:
calculator = my_torchani.models.ANI1ccx().to(device).ase()

# Now let's set the calculator for ``atoms``:
atoms.set_calculator(calculator)

# Now let's minimize the structure:
print("Begin minimizing...")
opt = BFGS(atoms)
opt.run(fmax=0.001)
print()


# Now create a callback function that print interesting physical quantities:
def printenergy(a=atoms):
    # Function to print the potential, kinetic and total energy.
    epot = a.get_potential_energy() / len(a)
    ekin = a.get_kinetic_energy() / len(a)
    print('Energy per atom: Epot = %.3feV  Ekin = %.3feV (T=%3.0fK)  '
          'Etot = %.3feV' % (epot, ekin, ekin / (1.5 * units.kB), epot + ekin))

# We want to run MD with constant energy using the Langevin algorithm
# with a time step of 1 fs, the temperature 300K and the friction
# coefficient to 0.02 atomic units.
dyn = Langevin(atoms, 1 * units.fs, 300 * units.kB, 0.2)
dyn.attach(printenergy, interval=1)
```

```
# Now run the dynamics:
print("Beginning dynamics...")
printenergy()
dyn.run(500)
```

Timing result:

```
# CPU
real        1m35.680s
user        1m26.560s
sys         0m9.126s

# GPU
real        0m18.612s
user        0m15.110s
sys         0m3.454s
```

3 Atomic environment vector

Using Behler-Parrinello symmetry functions (BPSF)

The original BPSF are used to compute an atomic environment vector (AEV) $\mathbf{G}_i^X$ for i-th atom of a molecule with atomic number X which are composed of M elements:

$$\mathbf{G}_i^X = \{G_1, G_2, G_3, \ldots, G_M\} \tag{1}$$

AEVs probe specific regions of an individual atoms's radial and angular chemical environment.

Each $\mathbf{G}_i^X$ is then used as input into a single NNP. Total energy of a molecule, E_T is computed from the outputs, E_i of the atomic number specific NNPs using:

$$E_T = \sum_{i}^{\text{all atoms}} E_i \tag{2}$$

Radial elements of $\mathbf{G}_i^X$:

$$G_m^{\mathrm{R}} = \sum_{j \neq i} \exp\left[-\eta \left(R_{ij} - R_{\text{shift}}\right)^2\right] f_{\mathrm{C}}\left(R_{ij}\right) \tag{3}$$

with piecewise cutoff function:

$$f_{\mathrm{C}}\left(R_{ij}\right) = \begin{cases} 0.5 \cos\left(\dfrac{\pi R_{ij}}{R_{\mathrm{C}}}\right) + 0.5 & \text{for } R_{ij} \leq R_{\mathrm{C}} \\ 0 & \text{for } R_{ij} > R_{\mathrm{C}} \end{cases} \tag{4}$$

The index m is over a set of η and R_{s}. Parameter η is used to change the width of the Gaussian distribution while the purpose of R_{s} is to shift the center of the peak. In an ANI potential, only single η is used to produce thin Gaussian peaks and multiple R_{s} are used to probe outward from the atomic center. The reasoning behind this specific use of parameters is two-fold:

- firstly, when probing with many small η parameters, vector elements can grow to very large values, which are detrimental to the training of NNPs.
- Secondly, using R_s in this manner allows the probing of very specific regions of the radial environment, which helps with transferability

Two modifcations are made to the original version of Behler and Parrinello's angular symmetry function to produce one better suited to probing the local angular environment of complex chemical systems.

- The first addition is θ_s, which allows an arbitrary number of shifts in the angular environment, and
- the second is a modified exponential factor that allows an R_s parameter to be added. The R_s addition allows the angular environment to be considered within radial shells based on the average of the distance from the neighboring atoms.

The effect of these two changes is that AEV elements are generally smaller because they overlap atoms in different angular regions less and they provide a distinctive image of various molecular features, a property that assists neural networks in learning the energetics of specific bonding patterns, ring patterns, functional groups, or other molecular features.

Given atoms i, j, and k angle θ_{ijk}, centered on atom i, is computed along with two distances R_{ij} and R_{ik}. A single element G_m^{Amod} of $\mathbf{G}_i^{\mathrm{X}}$ to probe the angular environment of atom i, takes the form of a sum over all j and k neighboring atom pairs, of the product of a radial and an angular factor

$$G_m^{\mathrm{Amod}} = 2^{1-\zeta} \sum_{j,k\neq i}^{\text{all atoms}} \left(1 + \cos(\theta_{ijk} - \theta_s)\right)^{\zeta} \times \exp\left[-\eta \left(\frac{R_{ij} + R_{jk}}{2} - R_s\right)^2\right] f_C(R_{ij}) f_C(R_{ik}) \tag{5}$$