Due date: Wednesday, December 08, 2021

!!Please make sure that you properly comment your code in the jupyter notebook!!

1 Neural network potential for Lennard-Jones clusters

In this worksheet, we develop a neural network (NN) potential for small Lennard-Jones (LJ) clusters. There will be three steps that will be worked on over the next three weeks:

(1) setting up the LJ clusters and creating several datasets (week 1)

(2) optimizing the hyper-parameters and training the NN (week 2)

(3) **application, transferability, and limitations (week 3)**

Each step will be submitted separately and the solution will be presented in the tutorial, so that it will be possible to work on subsequent tasks even if the solution to the previous task could not be found.

2 Monte Carlo sampling using the NN potential (10 pt)

We want to use a fitted NN to perform Monte Carlo (MC) sampling of the 3D clusters. Applying the NN potential in an actual simulation will require a certain accuracy and robustness of the potential. In particular, it is important to check that the created structures are still within the space the NN has been fitted in, as these potential are good for interpolation but, due to the lack of any physical meaning, they cannot be used for extrapolation.

In both the MC sampling and the NN fitting, random numbers are used. To make your results reproducible, initialize your random number generator with a specific seed (`np.random.default_rng(seed)`) for the MC sampling and specify a `random_state` in both `ShuffleSplit` for creating the training and test set and in `MLPRegressor` to initialize the weights.

2.1 Task 1: Fitting the initial NN

Use the dataset of the 3D LJ clusters at $T = 800$ K and fit an NN with 2 hidden layers and 10 nodes/layer, a logistic activation function, and an lbfgs solver. Split the data into 20 % test und 80 % training set. Report the R^2 score of the test data, and the MSE and MAE of the training and test datasets. One way to visualize the quality of the fit is to plot the predicted energy vs. the actual energy. For a 'perfect' fit, all data would fall on the diagonal line $y = x$. For your fit, plot the predicted energies as a function of the actual energies in a scatter plot together with the diagonal $y = x$. Make sure that the x and y axes have the same range and also the same dimension (so that you have a square plot).

2.2 Task 2: MC step function using the NN

Write a python function that performs an MC step, but instead of using the LJ potential to compute the energy, use the energy predicted by the NN potential. You can pass the `MLPRegressor` as a variable to the function and you need to compute the descriptors of the old and new configuration within the function. Make sure that you format the input and output values correctly!

Alternatively, you can use the `ase.calculators.calculator.Calculator` API to write a class that would work like the LJ calculator. To do so you need to inherit from `ase.calculators.calculator.Calculator` and implement `__init__` and `calculate` methods. The non working scafold of for this calculator looks like:

```python
from ase.calculators.calculator import Calculator, all_changes

def compute_distance_features(atoms):
    # COMPLEMENT
    return X

class SKlearnCalculator(Calculator):
    """Wrapper class to use a ML model from sklearn with ASE

    Parameters
    ----------
    model : class
        a trained sklearn model
    featurization : callable
        a function that takes the atomic structure and returns a set of
        features, e.g. internal distances
    """

    implemented_properties = ["energy"]
    "Properties calculator can handle (energy, forces, ...)"

    default_parameters = {}
    "Default parameters"

    nolabel = True

    def __init__(self, model, featurization, **kwargs):
        super(ASEMLCalculator, self).__init__(**kwargs)
        # COMPLEMENT

    def calculate(
        self,
        atoms=None,
        properties=["energy"],
        system_changes=all_changes,
    ):
```

```
        Calculator.calculate(self, atoms, properties, system_changes)
        # COMPLEMENT

        self.results["energy"] = energy
        self.results["free_energy"] = energy
```

2.3 Task 3: MC sampling using the NN potential

Now we want to perform MC sampling using the NN potential. We run the simulations for the 3D cluster at $T = 800$ K and a displacement $\sigma = 0.01$ for 30 000 MC steps. Every 2000 steps, print and record the energy predicted by the NN potential and the energy of the LJ potential. Plot both the NN and LJ energy for the configurations along the MC trajectory, that is as a function of the number of steps. Is the NN potential suitable to perform the MC sampling? Do the energies of the NN and LJ potential agree?

2.4 Task 4: Refitting the NN potential

Apparently, the NN potential fitted with the initial dataset is not robust enough to perform MC sampling as the MC moves create configurations that are outside the validity range of the potential. This is not uncommon and to resolve this problem, the NN potential can be refitted by adding additional configurations encountered during the sampling.

Use the MC sampling in the previous task with the NN potential and collect additional data every 20 steps. Sometimes, the NN can be really off and the LJ energy of the configurations extremely high. To avoid these artificial configurations, only collect data points if the LJ energy of a configuration is < 0. How many additional data did you create? Add the data to the original dataset and refit the NN. Report again R^2, MSE, and MAE, and plot the predicted energies vs the actual energies. Use the refitted NN to perform another MC sampling and again plot the energy of the configurations along the trajectory as given by the NN and LJ potential. Did the sampling improve?

Perform a second round of extending your dataset and refitting, and perform another MC sampling with the improved NN potential. Plot the energy of the configurations along the trajectory as given by the NN and LJ potential to evaluate the performance.

3 Transferability of the NN potential (10 pt)

We have already seen that the validity of the NN potential is restricted to the space covered by the training set. Here, we want to explore how transferable the NN potential is with respect to temperature and dimensions.

3.1 Task 1: Low and high temperature data

Use again a NN with 2 hidden layers and 10 nodes/layer, a logistic activation function, and an lbfgs solver. Fit the NN using the dataset of the 3D LJ clusters at $T = 10$ K with 20 % test and 80 % training data. Report the R^2 value, the MSE and MAE for the training and test set.

Use the NN to predict the energy of the dataset of 3D LJ clusters at $T = 800$ K, and report the R^2 value, the MSE and MAE for the dataset. Plot the predicted vs actual values in a square plot together with the diagonal for the test and training data at $T = 10$ K and for the dataset at $T = 800$ K. Can the NN fitted with data at the low temperature reliably predict the energies at the high temperature?

3.2 Task 2: Fitting with mixed datasets

To expand the transferability, the NN can be fitted using data created at different temperatures. For the 3D LJ clusters, in addition to the datasets at $T = 10$ K and 800 K, create datasets at $T = 50$ K, 200 K, and 500 K. Make sure to adjust the displacement to achieve acceptance ratios ≈ 0.5.

Combine the four datasets at $T = 10$ K, 50 K, 500 K, and 800 K and fit an NN (same architecture as above) with 20 % test and 80 % training data. Report the R^2 value, the MSE and MAE for the training and test set. Use the NN to predict the energy of the dataset of 3D LJ clusters at $T = 200$ K (which was not included in the training), and report the R^2 value, the MSE and MAE for the dataset. Plot the predicted vs actual values in a square plot together with the diagonal for the test and training data and for the dataset at $T = 200$ K. Can the NN fitted with data over a wide temperature range reliably predict the energies at $T = 200$ K?

3.3 Task 3: Transferability from 3D to 2D

By using data at different temperatures we could improve the transferability for the 3D LJ clusters. Use the 5 datasets at the different temperatures from the previous task and fit an NN (same architecture as above) with 20 % test and 80 % training data. Report the R^2 value, the MSE and MAE for the training and test set. Use the NN to predict the energy of the datasets of **2D** LJ clusters at $T = 200$ K and 2000 K, and report the R^2 value, the MSE and MAE for the datasets. Plot the predicted vs actual values in a square plot together with the diagonal for the test and training data and a separate plot for the datasets of the 2D clusters at $T = 200$ K and 2000 K. Can the NN fitted using the data collected for the 3D LJ clusters reliably predict the energies of the 2D clusters?

3.4 Task 4: Including all datasets

In a final fit, we combine all 5 datasets for the 3D LJ clusters and the 2 datasets for the 2D LJ clusters and fit an NN (same architecture as above) with 20 % test and 80 % training data. Report the R^2 value, the MSE and MAE for the training and test set, as well as separately for the entire datasets of the 2D LJ clusters at $T = 200$ K and 2000 K. Plot the predicted vs actual values in a square plot together with the diagonal for the test and training data and a separate plot for the datasets of the 2D clusters at $T = 200$ K and 2000 K. Can the NN fitted with all the data of both 3D and 2D LJ clusters reliably predict all the energies?