Due date: Wednesday, December 01, 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 Network architecture and weight optimization (10 pt)

We want to use the datasets created in the first part of the exercise to fit a NN that can predict the energy of the LJ clusters. As a first step we optimize the architecture of the NN (number of hidden layers, number of nodes/hidden layer, and activation function) and check the optimization of the weights with different solvers.

To solve the tasks, use the class `MLPRegressor` imported from `sklearn.neural_network`.

2.1 Task 1: Optimizer

The weights of the NN are initialized randomly. The same set of initial weights for a given architecture can be obtained by setting the variable `random_state`. In addition the variable `warm_start` can be used to reuse the solution of a previous fit.

Here, we want to compare the performance of different optimizers to fit the weights. Set `warm_start='false'` and use an NN with 2 hidden layers and 10 nodes/hidden layer. Fit the data for the 3D LJ cluster at $T = 800$ K using the 'adam' and the 'lbfgs' solver for two different sets of initial weights. Split the dataset into 80 % training and 20 % test data. To make sure the same set of initial weights is used for both optimizers, specify the `random_state` in the `MLPRegressor`. In total, there should be 4 different fits: pick two values for the `random_state` and optimize each of the two initial sets of weights with both the 'adam' and 'lbfg' solver. Compute and report the coefficient of determination (R^2) for the test dataset and the MSE and MAE for the training and test set. Report the number of iterations steps for each solver. Which of the two solvers performs better for the current problem?

2.2 Task 2: Hidden layer nodes

Determine how the MSE depends on the number of hidden layers and the nodes in the hidden layers. To do so, take again the dataset for the 3D LJ cluster at $T = 800$ K and split the dataset into 80 % training and 20 % test data. Use an 'lbfgs' solver and the 'logistic' activation function. Fit a series of NNs with one hidden layer and $1 - 100$ nodes in steps of 10, determine the MSE in the training and test set and plot the MSE as a function of the number of nodes. Use a logarithmic scale for the y-axis.

Fit another series of NNs with two hidden layers and $1 - 15$ hidden nodes/layer. Determine again the MSE in the training and test set and plot the MSE as a function of the number of nodes/layer. Use a logarithmic scale for the y-axis.

Which architecture (number of layers and nodes) would you select for the current problem?

2.3 Task 3: Activation function

As in Task 2, fit a series of NNs with one hidden layer and $1 - 100$ nodes in steps of 10 and two hidden layers and $1 - 15$ hidden nodes/layer, but instead of the 'logistic' activation function use the 'relu' activation function. Plot the MSE in the test and training set as a function of nodes. How do the results for the 'logistic' and 'relu' activation function compare?

3 Learning curve (10 pt)

3.1 Task 1: Size of training dataset

We want to determine the number of training data needed to converge the error in the test set. Use an NN with 2 hidden layers, 10 nodes/hidden layer, a logistic activation function, and an lbfgs solver. For each of the four datasets (2D LJ clusters at $T = 200$ K and 2000 K, 3D LJ clusters at $T = 10$ K and 800 K), select 20 % test data (200 data points). From the remaining 800 data points, randomly choose 20, 50, 100, 200, 300, 400, 500, 600, 700, 800 data points as training set. Fit an NN for each size of the training dataset and determine the MSE and R^2 in the test set. Plot the MSE and the R^2 as a function of the number of training data for each of the four datasets. What is the minimum number of training data needed to obtain a converged fit?