Hyperparameter Optimization of NN Models for a Single Water Molecule

Group 10

Data Preparation and Formulation Update

[*]New contents are marked in red to highlight.

Nicole Hu, Marissa Klee, John Pederson, Chengzhai Wang

Introduction

In this project, this group proposes to create a model interpreting and predicting the minimum-energy configuration of a simple system - a water molecule. We want to study the stable configuration of a water molecule. In this project, we will determine the minimum energy of the water using the trained neural network potential. When successfully demonstrating our model on a single water atom, we would want to investigate the hyperparameter optimization problem of the neural network potentials. We propose to apply simulated annealing and other local random search methods on hyperparameters to obtain better settings for neural network training. We would compare the performance of the implemented methods with available hyperparameter optimization packages on the market such as `hyperopt`[1].

Optimization Problem Definition

Problem 1

The first optimization problem we propose to study is to find the stable configuration of a water molecule by minimizing the potential energy. The first step is to build a neural network potential based on density functional theory calculations of a single water molecule at different bond lengths and bond angles. Details of the reference dataset can be found in section Data. The trained neural network potential ($f(\tilde{\mathbf{R}})$) maps the atomic coordinates ($\tilde{\mathbf{R}}$) to potential energy (E):

$$f(\tilde{\mathbf{R}}) = E$$

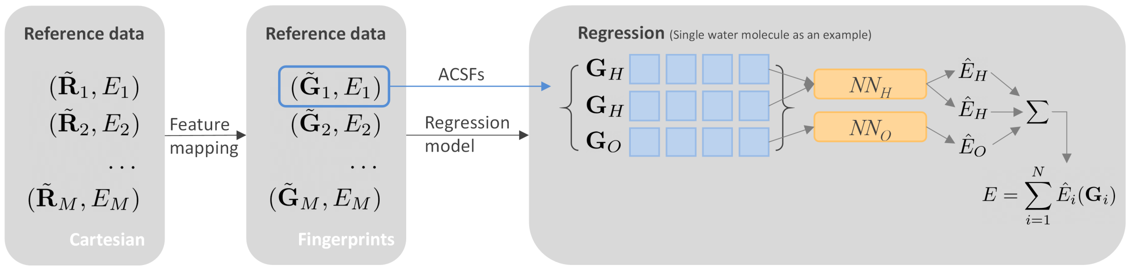
Reference data
$(\tilde{\mathbf{R}}_1, E_1)$
$(\tilde{\mathbf{R}}_2, E_2)$
$\dots$
$(\tilde{\mathbf{R}}_M, E_M)$
Cartesian
Feature mapping
Reference data
$(\tilde{\mathbf{G}}_1, E_1)$
$(\tilde{\mathbf{G}}_2, E_2)$
$\dots$
$(\tilde{\mathbf{G}}_M, E_M)$
Fingerprints
ACSFs
Regression model
Regression (Single water molecule as an example)
$\mathbf{G}_H$
$\mathbf{G}_H$
$\mathbf{G}_O$
NN_H
NN_O
$\hat{E}_H$
$\hat{E}_H$
$\hat{E}_O$
$\sum$
$E = \sum_{i=1}^{N} \hat{E}_i(\mathbf{G}_i)$

Figure 1: Flowchart of a Behler-Parrinello high-dimensional neural network to predict atomic energy contributions for a single water molecule.

We would perturb the atomic configurations of a water molecule by rattling the atoms away from their equilibrium bond length and bond angle. Originally we proposed to use `pyomo` to solve this non-linear problem (NLP) directly, but the current neural network potential implemented in `pytorch` does not have an interface with `pyomo`. Neural networks take an explicit mathematical form of weights and biases incorporated with a non-linear activation function. While it is theoretically possible to extract the weights and biases to express shallow neural networks and solve using exact global non-linear solvers, we failed to access the weights and biases in the trained surrogate model due to issues with the Python package and the high-dimensional structure of atomic neural networks. The neural network potential we constructed has two neural networks, one for oxygen atoms and another for hydrogen atoms, each having 3 layers and 20 nodes per layer. The non-linearity associated with the model would not be ideal to solve with traditional non-linear solvers, as implemented through `pyomo`. Therefore, we decide to implement heuristic and sampling-based optimization methods directly to energy minimization of the surrogate model.

Optimization Formulation: The objective is to minimize the approximated energy calculated by the neural network potential. The variables are the x, y, z positions of three atoms. This NLP problem has a formulation as shown below:

$$\min(f(\tilde{\mathbf{R}}))$$
$$\text{s.t. } \tilde{\mathbf{R}} \in \mathbb{R}_0^+$$

Problem 2

The second optimization problem we propose is to optimize the hyperparameter settings of the neural network potential for the chemical system of a single water molecule. The loss function of the neural network potential is defined below[2]:

$$\mathcal{L} = \frac{1}{N_{\text{struct}}} \sum_{i=1}^{N_{\text{struct}}} \left[\left(E_{\text{NN}}^i - E_{\text{Ref}}^i\right)^2 + \frac{\beta}{3 \cdot N_{\text{atom}}^i} \sum_{j=1}^{3N_{\text{atom}}^i} \left(F_{j,\text{NN}}^i - F_{j,\text{Ref}}^i\right)^2 \right]$$

where N_{struct} is the number of training images, E_{NN}^i is the predicted potential energy, E_{Ref}^i is the ground truth potential energy generated through DFT calculations, β is the force coefficient, N_{atom}^i is the number of atoms in the image, $F_{j,\text{NN}}^i$ is the predicted force component, and $F_{j,\text{Ref}}^i$ is the ground truth force.

Optimization Formulation: The training process of the neural network potential can be thought of as a black-box function ($g(\vec{x})$), which has the variables of multiple hyperparameters ($\vec{x}$) and yields the final error metrics of the surrogate model. Identified hyperparameters for this project include force coefficient, number of nodes, number of layers, learning rate, batch size, number of epochs, and activation functions. Numbers of nodes, layers, and epochs would be positive real integers. Force coefficient and learning rate would be positive reals. Activation functions are categorical variables. The type of the optimization problem would be complicated due to the nature of different types of activation functions, and therefore we do not define the type here. The optimization formulation here would be:

$$\min(g(\vec{x}))$$

In this problem, we would like to explore the optimization approach by integrating simulated annealing with some local search methods to optimize the hyperparameters. Instead of using existing solvers, we would focus more on local optimization algorithm implementations and make comparison with hyperparameter optimization software.

Data Description

The dataset for the optimization study consisted of a single water molecule with perturbed bond lengths and bond angles. The energies and forces are calculated via DFT. Quantum Espresso is used for DFT calculations with PBE exchange-correlation, 500 plane-wave cutoff, and K-point of (1, 1, 1). The total dataset consists of 400 images, 3 atoms per image.

Every data points has 3 features, the coordinates in 3D for every atom. The atomic coordinates are then used to calculate the corresponding atomic fingerprints as shown in Figure 1. Depending on the choice of parameters for atomic fingerprint generation, the number of features for the input of neural network model might vary. In this case, we chose a set of parameters that would yield a total of 80 dimensions.

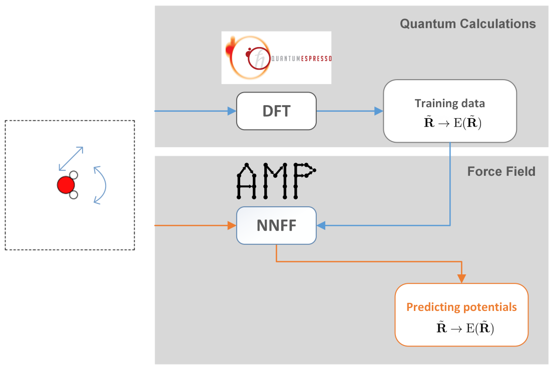

Figure 2: Schematic of data generation and training process in `AMP`[2] for neural network potentials. On the left hand side is a snapshot of a single water molecule in the training dataset.

Please refer to the attached Jupyter Notebook for both the initial input as the atomic coordinates, processed atomic fingerprints, and the corresponding potential energy of every image in the dataset.

Figure 3 plots the 2D potential energy surface mapped out by enumerating the O-H stretch and H-O-H bend while fixing the other O-H stretch at equilibrium bond length for better visualization.

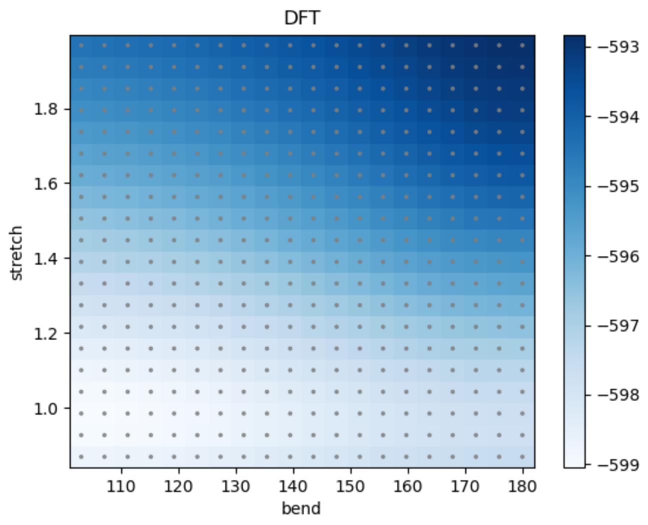

Figure 3: A schematic of the 2D potential energy surface mapped from H-O-H bend and O-H stretch as x and y axes from 400 training data points. The z-direction represented by the color map is the calculated potential energy in eV via DFT calculations. Training data are marked as gray points in the plot.

Preliminary ideas on optimization methods and solvers

1. For optimization problem 1, we would implement gradient-free optimization methods to find the optimal atomic configurations by minimizing the potential energy based on the trained neural network potential.

2. For optimization problem 2, we would implement simulated annealing and genetic algorithm to find a set of hyperparameters and implement various local random search methods to find better training results.

New contents below this line.

Optimization methods and solvers

Problem 1: Stable configuration with least potential energy

We manually implemented two gradient-free methods, direct search and genetic algorithm (GA). We compared the results to geometry optimization implemented through DFT calculations and optimization based on Nedler-Mead implemented in `SciPy`.

For the first optimization problem, we implemented a heuristic optimization method: the GA. This algorithm starts with a population of randomly generated individuals. Then the heuristic is used to improve the population using natural selection concepts: inheritance, mutation, crossover, and selection. For every generation, we rank the population of individual solutions by the potential energy calculated from surrogate model. We then select the individuals with the minimum energy before crossing over with bond angles and bond lengths. It will also mutate some of the solutions based on uniform distribution and record the best pairings. Since it's a stochastic process, we started with an ensemble of 20 genetic algorithms to generate an ensemble of solutions and took the mean of the best 20 solutions as our final results. A major advantage of GA is its compatibility with categorical variables, a particularly desirable feature for hyperparameter optimization. The main shortcoming, besides low speed, is that heuristic methods will not yield an exact solution but a solution in close proximity to the actual result. Below shows a pseudocode of GA:

Algorithm 1 Genetic algorithm (GA). NNP is short for neural network potential.

```
1: Generate a population of $N_{children}$ solutions randomly from the bound;
2: for each $k \in N_{genenration}$ do
3:     Rank population, $P_{k-1}$
           Compute $fitness(i)$ for each $i \in P_{k-1}$;        ▷ $fitness(i) = -NNP$
4:     Select $\chi$ best parents, $P_k$
           Based on ranked population, $P'_{k-1}$;
5:     Crossover
           Select $\gamma$ members of $P_k$; exchange variables as children; insert children into $P_k$
6:     Mutate
           Invert a randomly-selected variable based on $p_{mute}$;
7: end for each
```

For the second optimization problem, we implemented a positive-spanning basis direct search algorithm to sample over the fitted neural network and find the bond length and bond angle of the hydrogen atom in a water molecule which minimizes the total energy of the molecule. This optimization problem ought to serve to validate the efficacy of the neural network model, because the standard bond length and bond angle of water is well-attested in the literature, and to demonstrate the practical application of optimization on an atomistic neural network model. A direct search algorithm is applied to the trained surrogate neural network because its complex architecture does not allow for easy expression as a function. This neural network acts as a black-box, to be sure, but this black-box is much faster to evaluate than solving for the Kohn-Sham DFT equations, which is the original black-box problem. This direct search algorithm follows the principles of searching over a positive spanning set. The initial guess is taken to be the center of the first iteration of the algorithm, and $N+1$ other points are selected in the directions of the minimal positive spanning basis, where N is the dimensionality of the problem. This basis is considered to the be the unit vectors aligned with each positive axis plus a unit vector which is 45 degrees from each $N-1$ dimensional hyperplane composed of the negative axes. These unit vectors are scaled by some factor, a, and added to the center-point to get the remaining samples. All of these $N+2$ points are evaluated using the neural network model. If any sample point is found to fall outside of the input space feasible boundaries, such as 0 and 180 degrees or 0 Å in this case, its function value is set to positive infinity. The point which has the lowest function evaluation will then become the new center-point. If this point is the same as the previous center-point, then this means that the gradient over the potential surface is smaller than vector scaling factor, a. At that point, the vector scaling factor, a, will be reduced by an amount determined by the user. The algorithm will proceed until the vector scaling factor decreases below some pre-determined tolerance or until a maximum number of iterations has been reached.

Algorithm 2 Positive spanning set direct search Algorithm. NNP is short for neural network potential.

```
1: Determine an initial center-point $x_c$;
2: Calculate the set of $N+1$ vectors in the minimal positive span of the input space, $S \in \mathbb{R}^N$;
3: while $a > \epsilon$ & $i < N^{max}_{iterations}$ do
4:     Get points non-center points, $x_j \in X$
         Compute locations of non-center points by adding each vector $s_j \in S$ scaled by $a$ to the center-point;
5:     Evaluate points, $f(x)$, using NPP
         Set function value of points outside of input bounds to positive infinity;
6:     Compare
         Select $x_j$ with lowest function value to be the new center, $x_c$;
7:     Decrease $a$ by some amount $\delta a$;
         if $x_c^i = x_c^{i-1}$;
8: end while
```

Preliminary results

Problem 1: Stable configuration with least potential energy

We obtained preliminary results with manually implemented GA and direct search method. In Figure 4, we showed the evolution of the first GA solver with 10 generations in an ensemble to 20 GA solvers. Figure 5 tracks the convergence of the algorithm to a solution from a starting point of 120 degrees and 1.5 angstroms for the bond angle and bond distance, respectively.

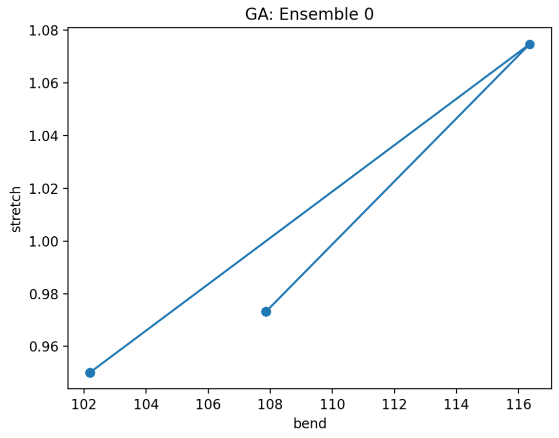

Figure 4: Best parents of 10 generations for one constituent in an ensemble of 20 GA solvers. Dots are overlaid because the best parents do not change over iterations.

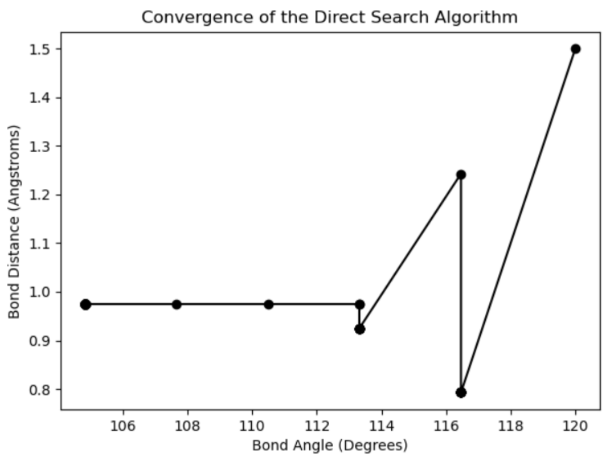

Figure 5: Center-points of the direct search algorithm as it converges to the ground state bond angle and bond distance from an initial guess of 120 degrees and 1.5 angstroms, respectively.

For better comparison, we created Table 1 with DFT geometry optimization results as ground truth, GA, direct search method and `SciPy` implementation of Nedler-Mead as a benchmarking system. Among current implementations, direct search method gives a solution closest to DFT calculations, and then is benchmarking Nedler-Mead, followed by GA method. Direct search method works well in this case because the 2D PES (Figure 3) does not have many local optima, and within tolerence, it is able to get very close to the accurate optimization results. The performance of GA is expected because heuristic methods like GA would yield results in close proximity to the actual solution, but cannot solve the problem exactly. However, the DFT geometry optimization results are still well within the bounds given by ensembles of GA optimizers.

Table 1: This table compares optimizations results from different methods. DFT geometry optimization is taken as ground-truth and is compared to different optimization methods based on surrogate models.

Method	Stretch	Stretch Error %	Bend	Bend Error %
DFT	0.9710	-	103.9690	-
GA	0.9820±0.0231	1.1329 ± 2.3790	107.4511±4.4482	3.3492±4.2784
Direct search	0.9747	0.3806	104.8326	0.8306
Nedler-Mead	0.9750	0.4119	102.6911	1.2291

Future Steps

Problem 1: Stable configuration with least potential energy

- Given the dataset is much more biased towards high-energy regions with longer stretch and extended bend (see Figure 3), we want to re-iterate the data generation and surrogate modeling construction step to refine our model with training dataset that have a better coverage of geometries near the equilibrium bond lengths and bond angle.

Problem 2: Neural network potential hyperparameter optimization

- After demonstrating that GA and direct search are both working, we want to piece them together to generate an algorithm to solve the hyperparameter optimization problem.

- Besides trying GA, we would also want to incorporate simulated annealing with other local search methods to compare the hyperparameter optimization results.
- We hypothesize that GA would offer a route to generate hyperparameter settings that are close to the best solution, and we want to further refine the search by applying local search methods.
- GA in this case would be ideal for its compatibility with categorical variables like the activation functions and discrete variables like number of layers and nodes.

References

[1] J. Bergstra, D. Yamins, and D. D. Cox. Making a Science of Model Search: Hyperparameter Optimizationin Hundreds of Dimensions for Vision Architectures. Technical Report 1, 2013.

[2] Alireza Khorshidi and Andrew A. Peterson. Amp: A modular approach to machine learning in atomistic simulations. *Computer Physics Communications*, 207:310–324, 10 2016.