Towards Accurate DeepMD Potentials for Aqueous Interfaces

Christian Loer Llemit

August 2024

Contents

Chapter 1

Introduction

1.1 Objectives

Describe importance of analyzing interfaces. Surface tension of different solutes.

Chapter 2

Theory

Interactomic Potential Energy Surface (PES) is an important property of the system that enables us to explain and predict materials properties such as interfacial energies, thermal expansion, cohesion, surface tension as well as chemical reactions. Representing PES accurately and efficiently is a major goal in molecular modelling. Existing approaches to PES modelling includes ab-initio models based on DFT, empirical potentials fitted with experimental data, and the relatively new machine learning based approach. The first one is accurate, parameter-free, and transferable but has bad scaling with system size. The second one is computationally efficient but has limited accuracy and transferability. The third one combines both the advantages of the two previous approaches in that it efficiently represents the PES for a wide variety of systems with the level of accuracy of ab initio quantum mechanics models and scales for large systems.
This work will focus on a particular scheme for molecular modelling, the deep neural network potential molecular dynamics.

2.1 Basics of Machine Learning

Before dealing with deep potential molecular dynamics, one must have a basic understanding on the principles of machine learning. Machine Learning (ML) falls under the umbrella term of Artificial Intelligence (AI) that focuses on developing algorithms that enable computers to learn from and make decisions based on data. It is based on the idea that systems can automatically improve their performance on a task through experience without human intervention. The algorithm will understand the data by deconstructing it in terms of hierarchy of concepts, with each concept defined through its relation to simpler concepts. This allows the algorithm to learn complicated concepts by building from simpler ones. Tasks such as classification, regression, transcription, clustering, translation, to name a few can be solved by ML.
Regression learning works by allowing the program to fit an input of dimension $\Re^n$ to an output of lower dimension $\Re^m$ by applying a mapping function f. Regression learning is an example of supervised learning since it involves training a model on a labeled dataset. In practice, the labeled data set is split into two: the training/learning and the testing/validation data. The first one is feed to the algorithm while the second one is used to validate the accuracy of the prediction.

Deep Learning is a type of ML that mimics the function of a neuron in the living systems. It is divided into layers that constitutes the input, output and multiple hidden layers where each layers contains collection of 'neurons' or nodes as schematically shown in Figure 2.1. An algorithm with one hidden layer is considered an Artificial Neural Network (ANN). Specifically, the algorithm is a subset of Regression learning that maps an input to an output. Deep forwarded network, also called feed-forward neural network, refers to the idea that information moves in only one direction from input, through the hidden layers, to the output. The goal of a feedforward network is to approximate the mapping $y = f(x; w)$ of input vector x to an output vector y by optimizing the values of the parameters/weights w. Networking works by applying composition of many different functions. For a network with N layers depth, N functions $f^1, f^2, \cdots, f^N$ will be chained in succession to form $f(x) = f^N(\cdots f^2(f^1(x)))$. The innermost function refers to the first layer while the outermost function refers to the output layer. Those functions that are in between refer to hidden layers since their outputs are not desired.

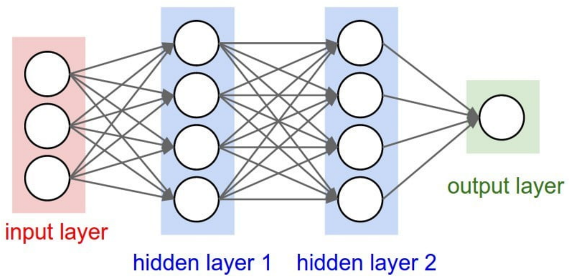

Fig. 2.1: Typical configuration of a Deep Learning algorithm with 2 hidden layers that maps a $\Re^3$ input vector to a scalar output vector.

To explain how each functions work, we look closely to the mathematical interpretation of a neuron, as schematically shown in Figure 2.2. For a given artificial neuron j, let there be $n+1$ inputs with signals x_0 through x_n and weights w_{0j} through w_{nj}. The x_0 input is assigned the value $+1$, which makes it a bias input with $w_{0j} = b_j$. There will be n actual inputs to the neuron: from x_1 to x_n. The inputs will be weighted sum and the result is used as an argument of the threshold function or activation function φ. Once the function reaches a threshold value θ_j or more, it will give the output or activation a_j of the jth neuron. Mathematically,

$$a_j = \varphi\left(\sum_{i=0}^{n} w_{ij}^T x_i\right) = \varphi(z_j) \tag{2.1}$$

Typical activation functions are Heaviside function, sigmoid, and rectified linear functions (ReLu).

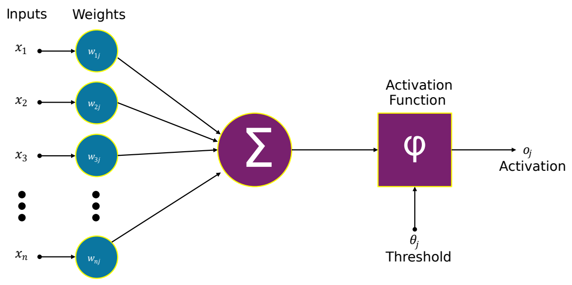

Fig. 2.2: The mathematical model of a neuron as first introduced by McCulloch and Pitts [1]. Image taken at commons.wikimedia.org/wiki/File:Artificial_neuron_structure.svg

One way to assess the performance of the training model is to compute the loss function, or cost function, $\mathcal{L}$ with respect to the weights. A simple choice is to compute the mean squared error (MSE) between the predicted $\hat{y}^{(\text{test})}$ and actual $y^{(\text{test})}$ value within the testing dataset.

$$\mathcal{L}^{(\text{test})}(w) = MSE^{(\text{test})} = \frac{1}{M}\sum_{i}^{M}\left(\hat{y}_i^{(\text{test})} - y_i^{(\text{test})}\right)^2 = \frac{1}{M}\sum_{i}^{M}\mathcal{L}_i^{(\text{test})} \tag{2.2}$$

where M is the number of test examples in the dataset. An algorithm should be introduced to minimize the error while allowing to gain experience by incorporating a training dataset. An intuitive way is to minimize the MSE on the training dataset with respect to the weights by carrying out the minimization problem called Gradient Descent

$$\nabla_w \text{MSE}_{\text{train}} \approx 0 \tag{2.3}$$

The weights are updated by subtracting a factor proportional to the gradient of the loss function.

$$w_{n+1} = w_n - \epsilon \boldsymbol{\nabla}_w \mathcal{L}(w) \tag{2.4}$$

where ϵ is the learning rate and n is the iteration step. Stochastic Gradient Descent extends the well-known Gradient Descent algorithm. Instead of computing the actual gradient, which is computationally expensive for large training datasets, it is replaced by an approximate gradient of a random subset of the training data. The loss function is minimized by taking the average gradient with respect to the weights for M' random minibatches of examples from the training set

$$\boldsymbol{\nabla}_w \mathcal{L}(w) \approx \frac{1}{M'}\sum_{i=1}^{M'} \boldsymbol{\nabla}_w L_i \tag{2.5}$$

where L_i is the loss function of a specific training example i.
The stochastic gradient descent does a good job in reducing the computational time, but the learning process can still be improved by implementing the backpropagation algorithm. The algorithm works by computing the gradient of the loss function with respect to each weights by utilizing chain rules in backward fashion, from last layer to the first.
Without going to full derivation, the idea is to measure the sensitivity of the loss function to variations in parameters such as weights, biases and the activations. For simplicity, consider the case of N layers with one neuron each. For one training example, the loss function will be $\mathcal{L}_0 = \left(a^{(N)} - y\right)^2$ where $a^{(N)}$ is the activation of the neuron at the last layer N and y is the actual value of the training example. From eqtn. (2.1), the activation will depend on the previous layer

$$a^{(N)} = \varphi\left(w^{(N)}a^{(N-1)} + b^{(N)}\right) \Longrightarrow \varphi\left(z^{(N)}\right) \tag{2.6}$$

where $w^{(N)}$ and $b^{(N)}$ are the weights and biases of the Nth layer. Taking relevant partial derivatives

$$\frac{\partial z^{(N)}}{\partial w^{(N)}} = a^{(N-1)}, \quad \frac{\partial z^{(N)}}{\partial b^{(N)}} = 1, \quad \frac{\partial z^{(N)}}{\partial a^{(N-1)}} = w^{(N)} \tag{2.7}$$

$$\frac{\partial a^{(N)}}{\partial z^{(N)}} = \varphi'\left(z^{(N)}\right) \tag{2.8}$$

$$\frac{\partial \mathcal{L}_0}{\partial a^{(N)}} = 2\left(a^{(N)} - y\right) \tag{2.9}$$

By Chain Rule,

$$\frac{\partial \mathcal{L}_0}{\partial w^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a^{(N)}} \frac{\partial a^{(N)}}{\partial z^{(N)}} \frac{\partial z^{(N)}}{\partial w^{(N)}} = 2\left(a^{(N)} - y\right) \varphi'\left(z^{(N)}\right) a^{(N-1)} \tag{2.10}$$

$$\frac{\partial \mathcal{L}_0}{\partial a^{(N-1)}} = \frac{\partial \mathcal{L}_0}{\partial a^{(N)}} \frac{\partial a^{(N)}}{\partial z^{(N)}} \frac{\partial z^{(N)}}{\partial a^{(N-1)}} = 2\left(a^{(N)} - y\right) \varphi'\left(z^{(N)}\right) w^{(N)} \tag{2.11}$$

$$\frac{\partial \mathcal{L}_0}{\partial b^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a^{(N)}} \frac{\partial a^{(N)}}{\partial z^{(N)}} \frac{\partial z^{(N)}}{\partial b^{(N)}} = 2\left(a^{(N)} - y\right) \varphi'\left(z^{(N)}\right) \tag{2.12}$$

To generalize to more than one neurons per layer, one must introduce subscript so that $a_j^{(N)}$ denotes the activation of jth neuron in the Nth layer while $w_{jk}^{(N)}$ is the weight that connects the jth neuron in the Nth layer and the kth neuron in the previous layer. Also, define $z_j^{(N)} = \sum_k w_{jk}^{(N)} a_k^{(N-1)} + b_j^{(N)}$. The chain rule is modified as follows

$$\frac{\partial \mathcal{L}_0}{\partial w_{jk}^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \frac{\partial a_j^{(N)}}{\partial z_j^{(N)}} \frac{\partial z_j^{(N)}}{\partial w_{jk}^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \varphi'\left(z_j^{(N)}\right) a_k^{(N-1)} \tag{2.13}$$

$$\frac{\partial \mathcal{L}_0}{\partial a_k^{(N-1)}} = \sum_{j=0}^{n_N-1} \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \frac{\partial a_j^{(N)}}{\partial z_j^{(N)}} \frac{\partial z_j^{(N)}}{\partial a_k^{(N-1)}} = \sum_{j=0}^{n_N-1} \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \varphi'\left(z_j^{(N)}\right) w_{jk}^{(N)} \tag{2.14}$$

$$\frac{\partial \mathcal{L}_0}{\partial b_j^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \frac{\partial a_j^{(N)}}{\partial z_j^{(N)}} \frac{\partial z_j^{(N)}}{\partial b_j^{(N)}} = \frac{\partial \mathcal{L}_0}{\partial a_j^{(N)}} \varphi'\left(z_j^{(N)}\right) \tag{2.15}$$

Eqtn. (2.13) can be rewritten by using eqtn. (2.14) for arbitrary layer l as

$$\frac{\partial \mathcal{L}_0}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \varphi'\left(z_j^{(l)}\right) \sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \varphi'\left(z_j^{(l+1)}\right) \frac{\partial \mathcal{L}_0}{\partial a_j^{(l+1)}} \tag{2.16}$$

This equation is what called the backpropagation formula where it got its name since the error of the previous layer depends on the values of the forward layers. This formula estimates the gradient that goes into updating the weights.

Deep learning continues to evolve rapidly, driven by advances in computational power, availability of large datasets, and innovative algorithms. It holds the promise of transforming various aspects of human life and industry by enabling more intelligent, autonomous systems.

2.2 Deep Neural Molecular Dynamics

Global Coordinate matrix

$$\mathcal{R} = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \ddots & \vdots \\ x_N & y_N & z_N \end{pmatrix} \tag{2.17}$$

Relative Coordinate Matrix with respect to atom i

$$\mathcal{R}^i = \begin{pmatrix} x_{1i} & y_{1i} & z_{1i} \\ x_{2i} & y_{2i} & z_{2i} \\ \vdots & \ddots & \vdots \\ x_{N_i i} & y_{N_i i} & z_{N_i i} \end{pmatrix} \tag{2.18}$$

Renormalized Coordinate Matrix

$$\tilde{\mathcal{R}^i} = \begin{pmatrix} s(r_{1i}) & \frac{s(r_{1i})x_{1i}}{r_{1i}} & \frac{s(r_{1i})y_{1i}}{r_{1i}} & \frac{s(r_{1i})z_{1i}}{r_{1i}} \\ s(r_{2i}) & \frac{s(r_{2i})x_{2i}}{r_{2i}} & \frac{s(r_{2i})y_{2i}}{r_{2i}} & \frac{s(r_{2i})z_{2i}}{r_{2i}} \\ \vdots & \vdots & \vdots & \vdots \\ s(r_{N_i i}) & \frac{s(r_{N_i i})x_{N_i i}}{r_{N_i i}} & \frac{s(r_{N_i i})y_{N_i i}}{r_{N_i i}} & \frac{s(r_{N_i i})z_{N_i i}}{r_{N_i i}} \end{pmatrix} \tag{2.19}$$

where $s(r)$ is a smooth switching function defined as

$$s(r) = \begin{cases} \frac{1}{r} & , r < r_s \\ \frac{1}{r}\left[x^3(-6x^2 + 15x - 10) + 1\right] & , r_s \leq r < r_c \\ 0 & , r \geq r_c \end{cases} \tag{2.20}$$

where $x = \frac{r - r_s}{r_c - r_s}$, r_c is the cutoff radius and r_s is the distance from atom i where smoothing is applied.

The Renormalized Coordinate matrix will serve as a descriptor that will feed into the ML algorithm for the training.

Chapter 3

Methodology

Initial trajectories were generated using Molecular Dynamics implemented in LAMMPS [2]. The pair potential used is based on the previously trained deep neural network potential (NNP) conducted by Sanchez-Burgos et al. [3].

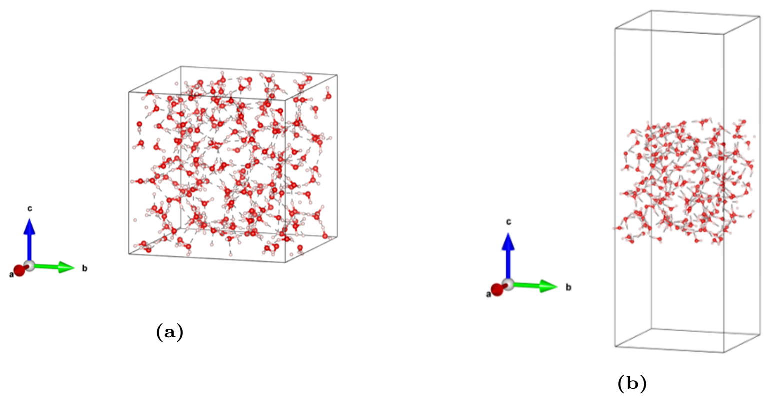

Fig. 3.1: Typical configurations for (a) bulk and (b) slab systems.

For the MD simulation, a system size of 192 water molecules was used for both the bulk and slab systems, as shown in Figure 3.1. The temperature was varied from 300 K to 600 K. The simulation profile of the bulk system is as follows: NPT ensemble at 1 bar for 2 ns, and a ramp from 1 bar to 10,000 bar for 10 ns. For the slab system, the simulation was done in NVT ensemble for 10 ns and the box size was based on the average length of the corresponding bulk system at a given temperature and at 1 bar. Vacuum was introduced as to create a total length of 50 Å in the direction normal to the interface. For both systems, a simulation step of 0.2 fs, thermostat relaxation time of 20 fs, and barostat relaxation time of 200 fs were applied. The surface tension was calculated according to Kirkwood-Buff equation [4] given by

$$\gamma = \frac{L_z}{2}\left[\langle P_{zz}\rangle - \frac{1}{2}\left(\langle P_{xx}\rangle + \langle P_{yy}\rangle\right)\right] \tag{3.1}$$

The trajectories were then labeled with their energy, force, and pressure tensor using Density Functional Theory (DFT) implemented in Quantum Espresso [5–7]. Strongly Constrained and Appropriately Normed (SCAN) exchange-correlation functional is widely used in the study of water systems due to its great predictability in describing hydrogen bonds and van der Waals interactions [8, 9]. However, for this study, the SCAN functional tends to be not numerically robust when implemented to systems with interfaces. Instead, this study tried to use accurate and numerically efficient r^{2}SCAN meta-generalized gradient approximation [10]. Optimized Norm-Conserving Vanderbilt pseudopotentials [11] were used with energy cutoff of 130 Ry and scf convergence threshold of 1×10^{-6} Ry.

The labeled frames were then feed into DeePMD-kit code [12–15] for the training of deep neural network potentials. A total of 7120 bulk frames and 2160 interface frames were used as training data. The training follows a typical two-neutral network architecture. First, atomic configurations are processed by the embedding network of three layers consisting of 25, 50 and 100 neurons each. The embedding follows the two-body smooth-edition scheme [16] that conserves radial and angular information within the cutoff radius of 6 Å. A switching function was applied for atoms beyond 5.5 Å to ensure smooth cutoff. Then, the atomic descriptors are build and given as input to a fitting network of three layers of 250 neurons each that outputs a scalar quantity such as energy. The parameters of the neural networks are set during the training procedure by minimizing loss function based on the mean squared error on the total energy and atomic forces predicted by the network with respect to the reference data. The iterative minimization was performed with a total of 4 million batches and a learning rate that exponentially decay from 1×10^{-3} to 3.51×10^{-8}.

Bibliography

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[2] A. P. Thompson *et al.*, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," *Comp. Phys. Comm.*, vol. 271, p. 108 171, 2022. DOI: `10.1016/j.cpc.2021.108171`.

[3] I. Sanchez-Burgos, M. C. Muniz, J. R. Espinosa, and A. Z. Panagiotopoulos, "A deep potential model for liquid–vapor equilibrium and cavitation rates of water," *The Journal of Chemical Physics*, vol. 158, no. 18, 2023.

[4] J. G. Kirkwood and F. P. Buff, "The statistical mechanical theory of surface tension," *The Journal of Chemical Physics*, vol. 17, no. 3, pp. 338–343, 1949.

[5] P. Giannozzi *et al.*, "Quantum espresso: A modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, 395502 (19pp), 2009.

[6] P. Giannozzi *et al.*, "Advanced capabilities for materials modelling with quantum espresso," *Journal of Physics: Condensed Matter*, vol. 29, no. 46, p. 465 901, 2017.

[7] P. Giannozzi *et al.*, "Quantum espresso toward the exascale," *The Journal of Chemical Physics*, vol. 152, no. 15, p. 154 105, 2020. DOI: `10.1063/5.0005082`.

[8] J. Sun, A. Ruzsinszky, and J. P. Perdew, "Strongly constrained and appropriately normed semilocal density functional," *Physical review letters*, vol. 115, no. 3, p. 036 402, 2015.

[9] M. Chen *et al.*, "Ab initio theory and modeling of water," *Proceedings of the National Academy of Sciences*, vol. 114, no. 41, pp. 10 846–10 851, 2017.

[10] J. W. Furness, A. D. Kaplan, J. Ning, J. P. Perdew, and J. Sun, "Accurate and numerically efficient r2scan meta-generalized gradient approximation," *The Journal of Physical Chemistry Letters*, vol. 11, no. 19, pp. 8208–8215, 2020, PMID: 32876454. DOI: `10.1021/acs.jpclett.0c02405`.

[11] D. Hamann, "Optimized norm-conserving vanderbilt pseudopotentials," *Physical Review B—Condensed Matter and Materials Physics*, vol. 88, no. 8, p. 085 117, 2013.

[12] H. Wang, L. Zhang, J. Han, and E. Weinan, "Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics," *Computer Physics Communications*, vol. 228, pp. 178–184, 2018.

[13] J. Zeng *et al.*, “Deepmd-kit v2: A software package for deep potential models,” *The Journal of Chemical Physics*, vol. 159, no. 5, 2023.

[14] D. Lu *et al.*, “86 pflops deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy,” *Computer Physics Communications*, vol. 259, p. 107 624, 2021.

[15] L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, *et al.*, “End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems,” *Advances in neural information processing systems*, vol. 31, 2018.

[16] L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, and W. E, “End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems,” in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.