Indicator configuration

Yonatan Kurniawan

1 Introduction

Mathematical models are important object in almost all scientific fields. They are used to summarize and explain some observed phenomena. In the development, we train the model $f(\theta)$ by tuning the parameters $\{\theta_n\}_{n=0}^{N-1}$ to best reproduce or fit a set of observed data $\vec{y} = \{y_m\}_{m=1}^{M}$. A common metric to measure the goodness of fit is weighted least square cost function,

$$C(\theta) = \frac{1}{2} \sum_{m=1}^{M} w_m \left(y_m - f_m(\theta)\right)^2, \tag{1}$$

where f_m is the m-th prediction point and w_m is the weight the m-th data point. If the data comes from a stochastic process, w_m is typically set to be the reciprocal of the standard deviation squared for data m. Using this cost function, the parameters are tuned to minimize the cost function.

In material science, models of interatomic potentials (IPs) have been widely used to study materials properties. IPs are commonly trained to fit atomic forces of several atomic configurations calculated from the DFT calculation, a training framework known as the force-matching method [1]. Other relevant quantities of choice, such as the lattice properties of the material, can also be added to the training data. The trained IPs are then used in a larger scale simulation to study macroscopical properties of the materials.

The training data can have different physical units, which gives a motivation to weigh the training non-uniformly. Even when the physical units are the same, e.g., when the training data only consist of the atomic forces, we might still want to apply non-uniform weights. For example, we want to weigh the data that corresponds to configurations with cracks higher if we are developing an IP that will be used to simulate cracks. In addition, choosing the weight values is an open question, and there has been some discussions about it [2] [3].

The advance of machine learning (ML) algorithm has pushed the development of machine learning interatomic potential (MLIP). MLIPs have been shown to be able to make accurate predictions of material properties, comparable to DFT calculations, while computationally cheaper, which allows accurate large-scale materials simulation. However, due to the lack of physics encoded in MLIPs, these models are awful in predicting materials properties that use atomic environments that have not been seen in the training set. This limitation motivates the active learning (AL) framework to maximize the volume of the domain and minimize the extrapolation distance. The basic idea of AL is to have a pool of unlabeled configurations (configurations without atomic forces values) and using some query function to pick the configurations that are the most beneficial. Then, the training data can be generated, e.g., via DFT calculation, to these important configurations. In a way, AL also reduce the computational time, especially by reducing the amount of expensive DFT calculation to do.

2 Indicator Configurations

In this section, we will explain how we use the FIM as a measure of information in the AL framework to find the important training configurations that we will refer to as indicator configurations. The idea of this framework is that if we have target predictions to make with our IPs as well as the target precision, i.e., error bars, it requires certain parameters of the IPs to be constrained by certain amounts. In other words, we need a certain information from the parameters to achieve the precision of the target predictions. When we train the IP to fit the training data, the training data give constrains on the parameters. Each training data, or training configurations, contains different information about the parameters, and thus constrain parameters differently. With the AL framework that we propose, we want to find combination of the training data, or configurations, that give use at least as much information about the IP parameters as required to achieve the target precision of the target predictions.

2.1 Background

Suppose that there are L atomic configurations available for training and there are M_ℓ data points, e.g. atomic forces, in configuration ℓ. We will also use $f_\ell(\theta)$ to represent the mapping from the IP parameters to the training quantities for configuration ℓ.

In this notation, we will write the cost function as

$$C(\theta) = \frac{1}{2} \sum_{\ell=0}^{L-1} w_\ell \sum_{m=0}^{M_\ell - 1} \left((y_\ell)_m - (f_\ell(\theta))_m \right)^2 , \tag{2}$$

where $(y_\ell)_m$ is the m^{th} data point in configuration ℓ, and similarly for the mapping $(f_\ell(\theta))_m$. We have also assumed that the data points in a single configuration are weighted equally and the weights only apply on the configuration level. [1]

As a measure of information, we use the Fisher information matrix (FIM), which quantifies the information that the data contain about the model parameters. In general, the FIM is defined as

$$\mathcal{I}(\theta) = E_Y \left[\frac{\partial^2 \log \mathcal{L}(\theta|\vec{y})}{\partial \theta^2} \middle| \theta \right] , \tag{3}$$

where $\mathcal{L}(\theta|\vec{y})$ is the likelihood of the model given the data and $E_Y[\cdot]$ denotes the expectation value over the probability of the data. For a least squares problem, the log-likelihood is proportional to the cost function. Thus, using Eq. (2) the FIM can be written as

$$\left(\mathcal{I}^{\text{conf}}(\theta)\right)_{ij} = \sum_{\ell=0}^{L-1} w_\ell \sum_{m=0}^{M_\ell - 1} \frac{\partial (f_\ell(\theta))_m}{\partial \theta_i} \frac{\partial (f_\ell(\theta))_m}{\partial \theta_j}, \tag{4}$$

where we have assumed that $E_Y \left[(y_\ell)_m - (f_\ell(\theta))_m \right] = 0$. We also denote the FIM that uses the training configurations as $\mathcal{I}^{\text{conf}}$ to differentiate it with the other FIM that will be discussed later. Notice that the second summation in Eq. (4) is the FIM of individual configuration, $\mathcal{I}_\ell^{\text{conf}}(\theta)$. Then, the FIM of the training configuration is given by [2]

$$\mathcal{I}^{\text{conf}}(\theta) = \sum_{\ell=0}^{L-1} w_\ell \mathcal{I}_\ell^{\text{conf}}(\theta). \tag{5}$$

[1] Because of this, if we use multiple quantities with different physical units, e.g., atomic forces and total energy, that correspond to the same configuration, we should index that configuration multiple times with different values of ℓ, each corresponding to different training quantity.

[2] We can compute $\mathcal{I}_\ell^{\text{conf}}(\theta)$ using the Jacobian of $f_\ell(\theta)$ by

$$\mathcal{I}_\ell^{\text{conf}}(\theta) = (J_\ell^{\text{conf}}(\theta))^T J_\ell^{\text{conf}}(\theta),$$

where

$$(J_\ell^{\text{conf}}(\theta))_{ij} = \frac{\partial (f_\ell(\theta))_i}{\partial \theta_j}.$$

In a similar way, the FIM of the model in predicting the target quantities of interest (QoI) is given by [3]

$$\left(\mathcal{I}^{\text{qoi}}(\theta)\right)_{ij} = \sum_{k=0}^{K-1} \frac{1}{\sigma_k^2} \frac{\partial(g(\theta))_k}{\partial\theta_i} \frac{\partial(g(\theta))^k}{\partial\theta_j}, \tag{6}$$

where K is the total number of the target QoI and $g(\theta)$ is a mapping from the IP parameters to the target QoI. The target error bars of the target QoI is given by σ_k.

2.2 Algorithm

Indicator configurations are found by tuning the weights in Eq. (5) so that the training configurations contain at least as much information as required to achieve the target precision of the target QoIs. As explained below, most of the weights will be zero, implying that the corresponding configurations are not needed in the training process. Note that this calculation will be done iteratively until the weight values converge. The algorithm for doing this calculation is shown in Algorithm 1. Now, let's discuss the step-by-step calculation in more details.

First, we choose an initial guess of the parameter values $\hat{\theta}$, and create a list of zeros to store the optimal weight values, denoted by $\vec{w}^{\text{opt}}$. Then we evaluate $\mathcal{I}^{\text{qoi}}(\hat{\theta})$ and $\mathcal{I}_\ell^{\text{conf}}(\hat{\theta})$ for all $\ell = 0, 1, \ldots, (L-1)$ configurations with these initial parameter values.

Then, using these FIMs, we will find a set of optimal weights via convex optimization. We conjecture that the needed information are contained in only a few training configurations. To reflect this conjecture, we set the objective function of the convex optimization to be [4]

$$O(\vec{w}) = \min_{\vec{w}} \|\vec{w}\|_1, \tag{7}$$

[3] We can also write Eq. (6) using the Jacobian of $g(\theta)$ as

$$\left(\mathcal{I}^{\text{qoi}}(\theta)\right) = (J^{\text{qoi}}(\theta))^T \Sigma^{-1} J^{\text{qoi}}(\theta),$$

where

$$(J^{\text{qoi}}(\theta))_{ij} = \frac{\partial(g(\theta))^i}{\partial\theta_j}$$

and Σ is a $K \times K$ diagonal variance matrix that contains the target precision of the target QoI.

[4] Note that since w_ℓ is constrained to be non-negative, the objective function Eq. (7) is thus the same as

$$O(\vec{w}) = \min_{\vec{w}} \sum_{\ell=0}^{L-1} w_\ell.$$

Algorithm 1 Indicator configuration algorithm

$\hat{\theta} \leftarrow$ initial_parameter_value ▷ Set an initial parameter values
$\vec{w}^{\text{opt}} \leftarrow \text{zeros}(L)$ ▷ Initialize a list to store optimal weights
while True **do**
    Compute $\mathcal{I}^{\text{qoi}}(\hat{\theta})$
    for all $\ell \in \{0, (L-1)\}$ **do**
        Compute $\mathcal{I}_{\ell}^{\text{conf}}$
    end for
    Solve convex optimization problem
        Objective: $O(\vec{w}) = \min_{\vec{w}} \|\vec{w}\|_1$
        Constraints: $w_\ell \geq 0, \sum_\ell w_\ell \mathcal{I}_{\ell}^{\text{conf}}(\hat{\theta}) - \mathcal{I}^{\text{qoi}}(\hat{\theta}) \succeq 0$
    $\vec{w}^{\text{nonzero}} \leftarrow \{w_\ell | w_\ell^{\text{dual}} < \text{tol}\ , \forall \ell \in \{0, (L-1)\}\}$ ▷ Get non-zero weights
    $\vec{w}^{\text{opt}} \leftarrow \{\max(w_\ell^{\text{opt}}, w_\ell^{\text{nonzero}}), \forall \ell \in \{0, (L-1)\}\}$ ▷ Update optimal weights
    if $\vec{w}^{\text{opt}}$ converged **then**
        break while loop
    else
        for all $\ell \in \{\ell | w_\ell^{\text{opt}} > 0.0\}$ **do**
            Generate training data for configuration ℓ
        end for
        $\hat{\theta} \leftarrow \arg\min_\theta C(\theta)$ ▷ Train the potential
    end if
end while

which pushes most of w_ℓ to zero. Additionally, we also want to constrain the weights such that

$$w_\ell \geq 0 \qquad \text{and} \qquad \sum_{\ell=0}^{L-1} w_\ell \mathcal{I}_\ell^{\text{conf}}(\hat{\theta}) - \mathcal{I}^{\text{qoi}}(\hat{\theta}) \succeq 0. \tag{8}$$

The significance of the first constraint is that weights are analogous to the inverse of squared standard deviation in statistical problem, so they are constrained to be non-negative. Then, the second constrain represents the goal that the information contained in the training configurations are at least as much as the required information for the target QoIs. Note that the inequality in Eq. (8) means that the matrix on the left-hand side is positive semidefinite. The interpretation of it is that we require the training configurations to contain at least as much information as needed by the target QoI.

From the result of convex optimization, we extract the weight values that are practically nonzero, $\vec{w}^{\text{nonzero}}$. Such weights tell us which configurations in the pool of configurations contain important information about the parameters that we need and how much we should weigh the configurations. In practice, we extract the weights that are practically nonzero by comparing the dual values of the weights, $\vec{w}^{\text{dual}}$ to the tolerance that we set in the convex optimization solver. The weights are practically non-zero when the corresponding dual values are below the solver's tolerance. Thus, the set of nonzero weights is given by

$$\vec{w}^{\text{nonzero}} = \left\{ w_\ell \middle| w_\ell^{\text{dual}} < \text{tol}, \forall \ell \in \{0, (L-1)\} \right\},$$

where tol is the tolerance that we set in the solver.

Having found the non-zero weights, we then update the list of optimal weights. For each configuration, we compare the corresponding weight from the convex optimization result with the value currently stored in the list of optimal weights and store the larger value between the two into $\vec{w}^{\text{opt}}$,

$$\vec{w}^{\text{opt}} = \{\max(w_\ell^{\text{opt}}, w_\ell^{\text{nonzero}}), \forall \ell \in \{0, (L-1)\}\}$$

This choice of updating $\vec{w}^{\text{opt}}$ ensures that the amount of information from the configurations in consecutive iterations is non decreasing. In this process, we also check if the optimal weights values converge within some tolerance, i.e., if the list of optimal weights significantly change from the update using the current convex optimization result. The calculation of indicator configurations can be terminated if $\vec{w}^{\text{opt}}$ has converged.

Step 4: Update parameters For the configurations with non-zero optimal weights, we generate training data, e.g., by running DFT calculation. Using this training data

and the optimal weights, we train the potential by minimizing the cost function in Eq. (2. Then, we repeat from step 1, but using the parameter values from the result of this training.

In the case where the optimal weights have not converged, we generate training data for the configurations that have nonzero optimal weights, for example by running DFT calculations. Note that there might be some configurations that already have reference data. For these configurations we can skip the data generating process. Then, using the reference data and the optimal weights, we train the potential to minimize the cost function in Eq. (2). We use the resulting optimal parameters to update $\hat{\theta}$ and repeat from the top of the loop, i.e., we compute the FIMs at these new parameter values and repeat the calculation until $\vec{w}^{\text{opt}}$ converges.

2.3 Additional notes

2.3.1 Precondition the FIMs

Solving the convex problem in Eq. (7) and (8) can be hard, for example when the elements of $\mathcal{I}^{\text{qoi}}(\theta)$ and $\mathcal{I}_\ell^{\text{conf}}(\theta)$ are different by several orders of magnitude. In this case, the convex optimization algorithm needs to take lots of iterations to explore a large region in the weight space to find the optimal weights.

A way to handle this situation is to precondition the FIMs. To do this, let

$$\tilde{\mathcal{I}}^{\text{qoi}}(\theta) = \frac{\mathcal{I}^{\text{qoi}}(\theta)}{\alpha} \qquad \text{and} \qquad \tilde{\mathcal{I}}_\ell^{\text{conf}}(\theta) = \frac{\mathcal{I}_\ell^{\text{conf}}(\theta)}{\beta_\ell}, \tag{9}$$

where α and β_ℓ act as some normalization constant to the FIMs. We suggest to use the Frobenius norm of the FIM for these constants. Using Eq. (9), we can modify the objective function of the convex optimization as

$$O(\vec{w}) = \min_{\vec{\tilde{w}}} \|\vec{\tilde{w}}\|_1, \qquad \text{where} \qquad \tilde{w}_\ell = \frac{\beta_\ell}{\alpha} w_\ell. \tag{7a}$$

The constraints are modified to be

$$\tilde{w}_\ell \geq 0 \qquad \text{and} \qquad \sum_{\ell=0}^{L-1} \tilde{w}_\ell \tilde{\mathcal{I}}_\ell^{\text{conf}}(\hat{\theta}) - \tilde{\mathcal{I}}^{\text{qoi}}(\hat{\theta}) \succeq 0. \tag{8a}$$

Note that solving Eq. (7a) and (8a) gives us a set of $\tilde{w}_\ell$. However, the weight values that are compared and stored in Step 3 are $w_\ell = \tilde{w}_\ell \alpha / \beta_\ell$.

2.3.2 Warm-up rounds

As with any other optimization algorithm, the optimization result can depend on the initial guess of the parameters. This is especially true for the indicator configurations algorithm presented in Sec. 2.2. Note that the optimal weights from the convex optimization depends the parameter values through the FIMs. The weights then affect the result of the optimal parameters from minimizing the cost, which is then used in the next iteration. In addition, convex optimization weights are also stored in the list of optimal weights, and it can affect the end result of the optimal weights.

As an example, suppose we start the calculation from a "bad" initial parameter values. As a result, the convex optimization weights might be very large. In the next iterations, after the parameter values move closer to the end result of the optimal parameters, the convex optimization weights might be smaller and more reasonable. However, the initial large weights will still be stored in the list of optimal weights and it might actually over-constrain the model. In contrast, if the initial parameter values are "good", that is, close enough to the end result of the optimal parameters, then the optimal weights will only be populated by the smaller and more reasonable values.

As a way to address this issue, we suggest having a warm-up rounds if a "good" initial parameters are unknown. The warm-up rounds have the same steps as laid out in algorithm 1, with the only exception is that we will not store the convex optimization weights in the list of optimal weights. The purpose of the warm-up rounds is to get the parameter values closer to the end result of the optimal parameters. After the warm-up rounds are over, then we continue with the normal iterations and start storing the optimal weights.

2.3.3 Regularization

Regularization term can be added in the cost function in Eq. (2). In Bayesian view, the regularization term can be thought as having a prior distribution about the parameters. As an example, adding an ℓ_2-norm of the parameters as the regularization, i.e., using ridge regression, correspond to having a Gaussian prior distribution on the parameters, while using the ℓ_2-norm, i.e., lasso regression, can be thought as using Laplace prior distribution.

Let us consider a case where we use an ℓ_2-norm regularization. For this case, the cost function that we use is

$$\tilde{C}(\theta) = C(\theta) + \gamma \frac{1}{2} \|\theta - \theta^0\|_2^2, \tag{10}$$

where θ^0 is the expectation values of the parameters from the Gaussian prior and γ is a Lagrange multiplier that control the strength of the regularization term. As an option, we choose γ such that least-squares cost function and the regularization term have comparable scales. The residuals in Eq. (2) are assumed to be independent to each other and normally distributed, i.e.,

$$\sqrt{w_\ell}\,(y_m - f_m(\theta)) \sim \mathcal{N}(0,1).$$

With this assumption, the least-squares cost function is distributed according to a chi-squared distribution with $(M-N)$ degrees of freedom and the expectation value of $(M-N)$. We also assume that in the prior distribution, $\{\theta_n - \theta_n^0\}_{n=1}^N$ are independent and normally distributed, such that

$$\frac{1}{2}\|\theta - \theta^0\|_2^2 \sim \chi^2(N),$$

with the expectation value of N. To make the scale of the least-squares cost function and the ℓ_2-norm regularization term to be comparable on average, thus we set

$$\gamma = \frac{M-N}{N}, \tag{11}$$

with a requirement that $M > N$.

2.3.4 Approximating the FIM

A bottleneck in algorithm 1 is the calculation of $\mathcal{I}^{\mathrm{qoi}}(\theta)$, since $g(\theta)$ is usually computationally more expensive than $f(\theta)$. However, we notice that after several iterations, that is, after the warm-up rounds (see Sec. 2.3.2), the parameter values start to converge. When this happens, $\mathcal{I}^{\mathrm{qoi}}(\theta)$ doesn't change significantly, assuming that $g(\theta)$ doesn't have some extreme behaviors. In this case, we can use some approximation, such as using Broyden's rank-1 update on the Jacobian matrix of $g(\theta)$, on the consecutive iterations to approximate $\mathcal{I}^{\mathrm{qoi}}(\theta)$. With this approximation method, Jacobian of $g(\theta)$ in the current iteration (iteration i) is

$$J_i^g(\theta^i) = J_{i-1}^g(\theta^{i-1}) + \frac{\Delta g^i - J_{i-1}^g(\theta^{i-1})\Delta\theta^i}{\|\Delta\theta^i\|_2^2}\Delta\theta^{iT}, \tag{12}$$

where $J_{i-1}^g(\theta^{i-1})$ is the Jacobian from the previous iteration, $\Delta\theta^i = \theta^i - \theta^{i-1}$ is the difference in the parameter values, and $\Delta g^i = g(\theta^i) - g(\theta^{i-1})$ is the difference in the model output. Note that this approximation requires one evaluation of $g(\theta)$.

As another alternative, we might be able to reuse $\mathcal{I}^{\text{qoi}}(\theta)$ for several iterations when the parameter values do not change significantly. However, this practice should be used with caution, noting that the precision of $\mathcal{I}^{\text{qoi}}(\theta)$ in the subsequent iteration will be sacrificed. Overall, we still recommend to use Broyden's rank-1 update if possible.

References

[1] F. Ercolessi and J. B. Adams. Interatomic potentials from first-principles calculations: The force-matching method. *EPL (Europhysics Letters)*, 26(8):583, Jun 1994.

[2] Michael R. Fellinger, Hyoungki Park, and John W. Wilkins. Force-matched embedded-atom method potential for niobium. *Physical Review B*, 81(14):144119, Apr 2010.

[3] Thomas J. Lenosky, Joel D. Kress, Inhee Kwon, Arthur F. Voter, Byard Edwards, David F. Richards, Sang Yang, and James B. Adams. Highly optimized tight-binding model of silicon. *Physical Review B*, 55(3):1528–1544, Jan 1997.