Solving Quantum Many-Body Problems with Neural Networks

Jane Kim
(Dated: October 5, 2019)

This project aims to enhance the variational Monte Carlo method with machine learning to solve for the ground state energy of a system of bosons. The trial wave function is replaced by a flexible feedforward neural network of one hidden layer. The natural cost function for a variational problem is the energy of the system, which we will compute stochastically and minimize with respect to the weights and biases of the network. The particular model of choice is exactly solvable, so the accuracy of the final result for the ground state energy and wave function can be measured quantitatively.

I. INTRODUCTION

The wave function Ψ of an isolated quantum system is a mathematical quantity that depends on its degrees of freedom and fully characterizes its state. The amount of information that needs to be encoded in the wave function grows exponentially with the number of particles in the system, posing a difficult computational problem for our limited classical resources. Typical methods that rely on calculating the wave function attempt to either compress the quantum state efficiently or sample a finite number of particle configurations that contribute the most to the description of the system. The goal of this project is to combine both approaches by using machine learning to enhance a traditional variational Monte Carlo calculation.

II. CALOGERO MODEL

We will use this hybrid method to calculate the ground state energy of a simple 1D quantum system of N bosons. In the Calogero model, the bosons are trapped in a harmonic oscillator potential and interact via a pair-wise inverse squared potential. The Hamiltonian, or energy operator, of the system can be written as

$$\hat{H} = \sum_{i=1}^{N}\left(-\frac{1}{2}\frac{\partial^2}{\partial x_i^2} + \frac{1}{2}x_i^2\right) + \sum_{i<j}^{N}\frac{\nu(\nu-1)}{(x_i - x_j)^2}, \quad (1)$$

where x_i is the position of the ith particle, ν is an interaction parameter, and the constants $\hbar, m, \omega$ are set to one for convenience. Then the energy of the system in the state $\Psi(\boldsymbol{x})$ is given by

$$E = \frac{\int \Psi^\dagger \hat{H}\Psi d\boldsymbol{x}}{\int \Psi^\dagger \Psi d\boldsymbol{x}} \equiv \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \quad (2)$$

The most probable state for any isolated quantum system is the ground state. So it is of particular interest to calculate the ground state wave function Ψ_0 that produces the lowest possible energy E_0 using equation (2). For our system of bosons, these quantities can be found analytically and are given by

$$\Psi_0^{exact}(\boldsymbol{x}) = \exp\left(-\frac{1}{2}\sum_{i=1}^{N}x_i^2\right)\prod_{i<j}^{N}|x_i - x_j|^\nu, \quad (3)$$

and

$$E_0^{exact} = \frac{N}{2} + \frac{\nu}{2}N(N-1). \quad (4)$$

Since this system is exactly solvable, we will be able to quantitatively measure the performance of our algorithm.

III. VARIATIONAL MONTE CARLO

Variational methods are built on the variational principle, which states that for *any* guess for the ground state wave function Ψ, the expectation value of the energy is an upper bound for the true ground state energy:

$$E_0 \le E = \frac{\langle\Psi|\hat{H}|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \quad (5)$$

This integral is typically very high dimensional, so it is calculated using Monte Carlo integration.

For this principle to be useful, we need to provide a very good guess for the ground state wave function. The traditional approach would be to use physical intuition of the problem to design a parametrized trial wave function $\Psi_T(\boldsymbol{x};\boldsymbol{\alpha})$ that has expected limiting behaviors based on theory or a form inspired by experiment. By allowing the wave function to be "flexible", we can find a better upper bound for the ground state energy by minimizing the energy with respect to the parameters $\boldsymbol{\alpha}$ of the wave function.

An issue with this approach is that it heavily relies on a physicist's insight. If the trial wave function is parametrized in the wrong way, it will not provide a decent upper bound for the ground state energy. It also lacks broad applicability to systems we have not yet explored, because each new system needs a new, physically motivated guess for the ground state wave function.

IV. UNIVERSAL APPROXIMATION THEOREM

What we need to overcome the limitations of the traditional variational Monte Carlo method is a maximally flexible trial wave function that can be applied to any problem. Luckily, machine learning may provide the solution. The universal approximation theorem states that a feedforward neural network of just one hidden layer can approximate any continuous function, provided there are enough hidden neurons. This suggests we can use such a network to represent our trial wave function, and minimize the energy with respect to the weights and biases of the network. This will be shown more explicitly in section VI.

V. DATA

The data we will be using are generated during the calculation of the energy E and its gradient with respect to the parameters of the network $\frac{\partial E}{\partial \boldsymbol{\alpha}}$. Both of these quantities are computed stochastically at each iteration, meaning we approximate them by averages over samples of particle configurations pulled from some distribution. For example, we can cleverly shuffle things around in the integral in (2) to estimate the energy by a simple average over "local energies"

$$E \approx \frac{1}{n}\sum_{k=1}^{n} E_L(\boldsymbol{x}_k) \equiv \langle E_L \rangle, \tag{6}$$

where n is the number of samples and $\boldsymbol{x}_k$ is the kth sample of particle positions. Since the square of the normalized wave function defines the probability of finding the particles in a certain configuration, we will be pulling the random samples from the distribution

$$P(\boldsymbol{x};\boldsymbol{\alpha}) = \frac{|\Psi_T(\boldsymbol{x};\boldsymbol{\alpha})|^2}{\langle \Psi_T | \Psi_T \rangle}. \tag{7}$$

Neural networks, as well as other standard machine learning algorithms, usually start with a data set that is split into training, testing, and validation sets. Then the parameters of the network are tuned to minimize some cost function using the training data alone. In our case, the particle configurations are generated during each iteration and depend on the parameters of the network themselves. We do not want to waste memory by storing all of the positions since we just need to keep running sums for the energy and its gradient. The accuracy of the averages also improve with a larger number of samples. So all of the data will be used for training and the cost function is the average energy (6).

VI. ALGORITHM

Many of the elements of a variational Monte Carlo calculation overlap with the elements of a typical machine learning algorithm. They both assume a set of random parameters, measure how wrong the result is based on the assumption, and have some way of changing the parameters in order to get a better result. In one iteration of our hybrid algorithm, we will estimate the energy and its gradient using a predefined number of samples n. Then stochastic gradient descent is used to update the parameters until the gradient is smaller than some chosen tolerance ϵ. Both the energy and the gradient will be used to track the progress of the optimization process.

We will represent the trial wave function of our system of bosons as a feedforward neural network with one hidden layer. The inputs are the positions of the N one-dimensional bosons and we can choose our network to have M hidden neurons. We will notate the input layer and hidden layer as $\boldsymbol{x} \in \mathbb{R}^N$ and $\boldsymbol{h} \in \mathbb{R}^M$, respectively. They are fully-connected by weights $W \in \mathbb{R}^{M\times N}$ and are biased by $\boldsymbol{b} \in \mathbb{R}^M$, so that

$$\boldsymbol{h} = W\boldsymbol{x} + \boldsymbol{b}. \tag{8}$$

A general wave function is complex, but for a system of bosons, it is positive everywhere. This is convenient, because our network only needs to have one output neuron u, instead of two to represent the real and imaginary parts. The output of the network is given by

$$u = \boldsymbol{w}^T \boldsymbol{f}(\boldsymbol{h}), \tag{9}$$

where f is the activation function and the vector $\boldsymbol{w} \in \mathbb{R}^M$ contains the weights connecting the hidden neurons with the single output. We have bolded the activation function in (9) to emphasize that the result is a vector, with the function applied to each element of $\boldsymbol{h}$ separately. Finally, we define the trial wave function to be

$$\Psi_T(\boldsymbol{x}; W, \boldsymbol{w}, \boldsymbol{b}) = \exp(u) = \exp(\boldsymbol{w}^T \boldsymbol{f}(W\boldsymbol{x} + \boldsymbol{b})). \tag{10}$$

VII. PARAMETERS

The inverse squared potential pushes the bosons apart for $\nu < 0$ and $\nu > 1$ and pulls them together for $0 < \nu < 1$. It may be interesting to see how the network performs for different values of the interaction parameter ν. If the training of the network proves to be difficult, we may need to introduce the interaction potential more gradually. Assuming the changes in the network parameters are small for small changes in ν, we can first train the network for the non-interacting case then slowly increase ν to our desired value, optimizing the parameters as we go.

There are several additional hyperparameters we can tune to improve the performance of our algorithm. For example, we can explore how many hidden neurons M are required to well approximate the ground state, and the learning rate η for stochastic gradient descent can be optimized to decrease training time. We can also determine the best tolerance ϵ for stopping the optimization process, and the best number of samples n to accurately and efficiently approximate the energy and its gradient.

VIII. POSTER

Since this neural network will be trained in an unconventional way, I plan on writing my code from scratch in Python. For my poster, I intend on describing the structure of my code for the calculation in detail and displaying the relevant math in LaTeX. I will use Keynote to make the poster itself, and print it at the MSU Main Library. For maximum efficiency, the training will be performed on the HPCC cluster at ICER.