Machine Learning Quantum Many-Body Systems Using Neural Autoregressive Distribution Estimation

Authors: Caleb Sanders,* Ignacio Varela,* Zhexuan Gong,† and Alex Lidiak†

February 2021

Abstract

1 Introduction

Accurately modeling and solving interacting many-body systems is a very useful and challenging endeavor in quantum mechanics. Specifically, obtaining the ground state energy of a quantum many-body system has various applications in condensed matter, chemistry, nuclear, and other areas of physics[1]. However, a persistent problem in the field of quantum mechanics is dealing with the exponentially increasing dimensionality of these many-body systems. The number of possible spin states in a quantum system of N qubits increases as d^N (where d is the number of possible discrete quantum values per qudit) with its wavefunction given by

$$|\psi\rangle = \sum_{s}^{2^N} \psi_s(\Omega)\,|s\rangle\,, \tag{1}$$

where $|\psi\rangle$ is a linear superposition of basis states $|s\rangle = |s_1, s_2 ... s_N\rangle$. As N grows, writing the many-body wavefunction given by Eq 1 becomes exponentially inefficient. Similarly, the dimensionality of both the Hamiltonian matrix and the computational cost required to compute it's eigenenergies also increases as d^N, making these calculations very difficult for large N. Today, it is infeasible to classically compute these ground states for systems of more than about 30 particles on most computers, with the upper limit generally agreed to be 51 particles when using the most powerful HPCs[2]. Thus, with large systems, we introduce the need for an approximation of their wavefunction, also known as an "ansatz". Ideally, this compact representation of the wavefunction should have a number of parameters that scales polynomially rather than exponentially. In many cases, it is possible to obtain such an approximation because the the probability distribution of states in physical many-body systems only occupies a small region of the full Hilbert space.

Over the years, many ansatze have been derived (mean-field, Jastrow, matrix product state, etc.). While these ansatze approximate the many-body wavefunction with fewer parameters, they have limitations in their ability to simultaneously model high entanglement and variational freedom [1]. However, neural networks have recently shown great promise in representing entangled many-body wavefunctions and have opened doors to new areas of research. In the neural network approach, a network is established with an input layer of N nodes, a hidden layer, and an output layer representing $\psi(s)$. If the hidden layer is constructed with a reasonable number of nodes (on the order of N), then the number of parameters (corresponding to the network weights and biases) scales polynomially. Furthermore, this approach introduces nonlinear mapping between inputs and outputs, thus allowing entanglement correlations to be encoded into the network.

Finally, modern machine learning tools such as backpropagation, gradient descent algorithms, and GPU parallelization introduce the potential for a fully

integrated system that efficiently optimizes a network to the ground state energy of an arbitrarily large many-body system. In this report we investigate the use of a feed forward neural network (FFNN) as an ansatz and introduce a proof of concept optimization system.

2 Background

A feed forward neural network (FFNN) is one of the most commonly used types of neural network. Figure 1 shows an example diagram of a simple FFNN.

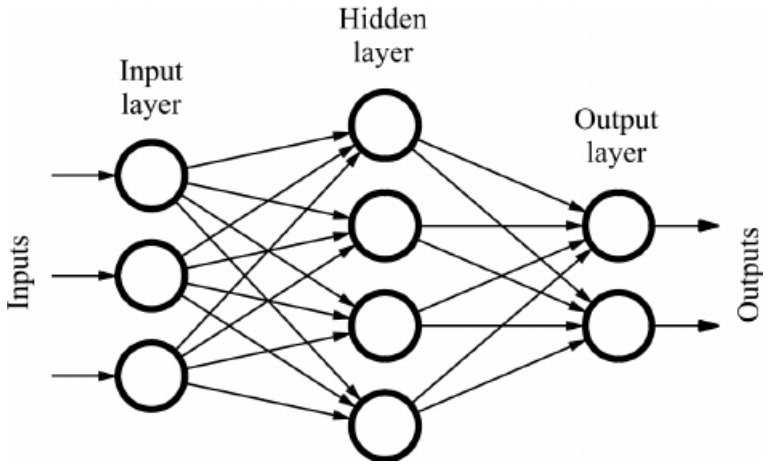

Figure 1: Example FFNN architecture diagram

Simple FFNNs typically consist of 3 layers: an input layer, a hidden layer and an output layer. Connections are only present between adjacent layers, with no intra layer connections. Normally, the layers are fully connected, meaning each node in one layer is connected to every node in the next. Information in an FFNN flows in one direction from input to output, hence the name. Each node and connection is represented by a number value, the range of which can be tailored to different applications. The value of a given node a_i can be expressed as a function of the nodes feeding into it and the connection values, known as weights. Each node is multiplied by its corresponding weight w_{ij} and are summed, meaning a_i is a linear combination of all the nodes in the previous layer. A bias vector b (not shown) length equal to that of a_i is added after the product. Finally, an activation function σ is included to 1)introduce nonlinearity to the system and 2) control the range and distribution of possible values for a_i. Equation **??** shows the function for a given layer, where W is the weight matrix for the given layer and s is the vector of values for the previous layer.

$$a = \sigma(W \bullet s + b) \tag{2}$$

In our network, the input consists of a simple 1-dimensional vector s of binary values with length N, corresponding to the state of each bit in the system. The output vector is a vector of length d, where d is the number of possible discrete quantum values for each qubit. The two values in the output vector correspond to the wavefunction coefficient for a given bit s_i occupying the up or down state.

$$\ln \psi \tag{3}$$

3 Implementation

3.1 Neural Autoregressive Distribution Estimation (NADE)

For large N, summing across all possible spin states to calculate the energy gradient is computationally impossible, thus requiring a stochastic sampling method to draw from a distribution of d^N states. NADE is an exact probability density estimation algorithm that naturally lends itself to parallelization, as samples can be generated independent of each other. In comparison to Markov-chain Monte Carlo (MCMC) sampling, NADE is much more computationally efficient and also produces broader spin configuration samples of the full Hilbert space [1]. When generating samples from a distribution, we first commit to representing the probability of sampling a given state $P(s)$ as a product of conditional probabilities,

$$P(s_1, ..., s_N) = \Pi_{i=1}^{N} p_i(s_i|s_{i-1}, ..., s_1) \tag{4}$$

as proposed by Uria et. al. The autoregressive nature of this sampling procedure allows us to decompose the probability of obtaining a bit at site i into the product of conditional probabilities of selecting all bits prior to site i. Furthermore, the outputs of a forward pass through the network are denoted by

$$v_i = (v_{i,s_1}, v_{i,s_2}, ..., v_{i,s_M}) \tag{5}$$

where v_i is an un-normalized probability distribution for s_i to take one of M discrete possible quantum values [1]. While v_i can be expanded to include both real and complex values (as our collaborator Alex Lidiak has shown), we have chosen to only include real values.

For the autoregressive property to be properly enforced, each v_i must only depend on spins $s_1, ..., s_{i-1}$. This condition can be easily met by applying a mask to the input before passing it through the network. To generate v_i for a bit at site i, values $s_1, ..., s_{i-1}$ of the mask are set to 1 while values $s_i, ..., s_N$ are set to 0. It is important to note that, to sample s_1, a fully masked state (state of all zeros) must be passed through the network. Thus, the first sampled bit is solely dependent on the the network's parameters at the time it is sampled. Once v_i is generated for a given state, the distribution is normalized according to the l_1 norm,

$$p_i(s_i|s_{i-1}, ..., s_1) = \frac{exp(v_{i,s_i})}{\sum_{s'} |exp(v_{i,s'})|} \tag{6}$$

In the same way this sampling method is used to produced conditional probabilities for sampling a given state, it can also be used to produce conditional wavefunctions in a quantum system. In such a system, $\psi(s)$ is given by

$$\psi(s) = \prod_{i=1}^{N} \psi_i(s_i|s_{i-1}, ..., s_1). \tag{7}$$

The only difference between dealing with conditional probabilities and conditional wavefunctions is, in the $\psi(s)$ picture, v_i is normalized according to the l_2 norm since the probability of obtaining a given state is defined by the Born rule, $P(s) = |\psi(s)|^2$. Once v_i has been obtained for a given state, v_{i,s_1}^2 can be denoted as the probability of sampling a positive bit while v_{i,s_2}^2 can be denoted as the probability of sampling a negative bit. The positive-bit probability can then be used to generate a sampled bit using the Bernoulli method (note, Bernoulli can only be used in systems where $d = 2$). Figure 3.1 displays an example sample distribution generated using NADE sampling for a system size of 5 qubits.

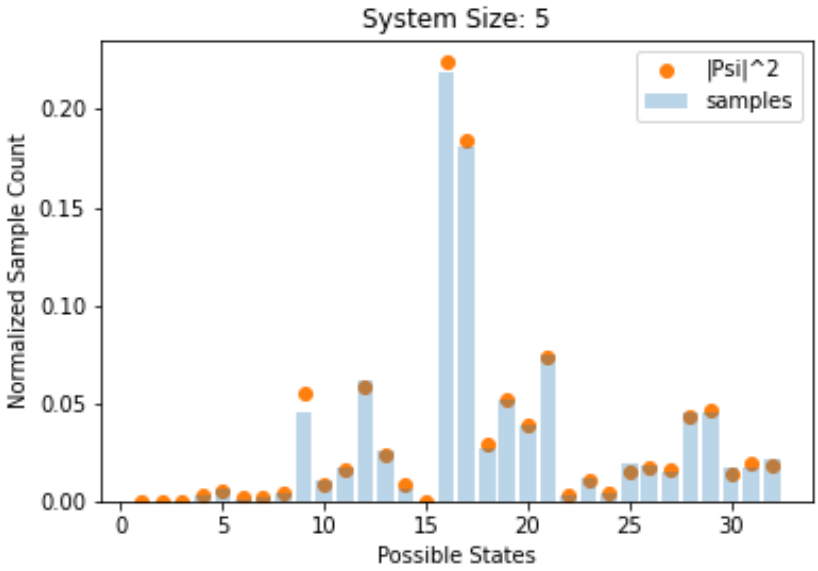

Figure 2: A distribution of 2500 samples generated using NADE for 5 qubits. Normalized sample count is plotted alongside theoretical sample probability for 2^5 possible states

3.2 Energy Gradient Calculation

3.3 Optimization

4 Results

5 Equations and Derivations

5.1 Autoregressive Psi Gradient

$$\frac{\partial \psi(s_1, ..., s_N)}{\partial \Omega_k} = \prod_{i=1}^{N} \frac{\partial \psi_i(s_i|s_{i-1}, ..., s_1)}{\partial \Omega_k} \tag{8}$$

$$\frac{\partial \log \psi(s_1, ..., s_N)}{\partial \Omega_k} = \sum_{i=1}^{N} \frac{\partial \log \psi_i(s_i|s_{i-1}, ..., s_1)}{\partial \Omega_k} \tag{9}$$

$$\frac{\partial \log \psi_1(s_1)}{\partial \Omega_k} = \frac{\partial \psi_1(s_1)}{\partial \Omega_k} \left(\frac{1}{\psi_1(s_1)} \right) \tag{10}$$

$$\frac{\partial \log \psi(s_1, ..., s_N)}{\partial \Omega_k} = \sum_{i=1}^{N} \frac{\partial \psi_i}{\partial \Omega_k} \left(\frac{1}{\psi_i} \right) \tag{11}$$

5.2 Energy

$$E = \langle \varepsilon(s) \rangle = \sum_s |\psi(s)|^2 \varepsilon(s) = \psi^\dagger \varepsilon \psi \tag{12}$$

$$E = \frac{\sum_(k = 1)^k \varepsilon(s_k)}{k} \tag{13}$$

$$\varepsilon(s) = \sum_{s'} H_{ss'} \frac{\psi(s')}{\psi(s)} \tag{14}$$

$$s'_i = \sigma_i^x s \tag{15}$$

5.3 Energy Gradient

$$\frac{\partial E}{\partial \Omega_k} = \left\langle 2Re\left(\varepsilon(s) - E \right) \frac{\partial \ln \psi(s)}{\partial \Omega_k} \right\rangle \tag{16}$$

$$\frac{\partial E}{\partial \Omega_k} = \sum_K \varepsilon(s) \frac{\partial \ln \psi(s)}{\partial \Omega_k} - \frac{E \sum_K \frac{\partial \ln \psi(s)}{\partial \Omega_k}}{K} \tag{17}$$

5.4 TFIM

$$H = \sum_{i-1}^{N-1} \sigma_i^z \sigma_{i+1}^z + B \sum_i \sigma_i^x \tag{18}$$

Ising:

$$\varepsilon(s) = \frac{1}{\psi_s^*} \sum_{s'} \psi_{s'}^* H_{ss'} \tag{19}$$

Transverse:

$$\varepsilon(s) = \frac{\psi_{s'}^*}{\psi_s^*} B = B \frac{\sum_i \psi_{s'}(i)}{\psi_s} \tag{20}$$

Both:

$$\varepsilon(s) = \sum_{i-1}^{N-1} s_i s_{i+1} + \frac{B}{\psi_s} \sum_i \psi_{s'}(i) \tag{21}$$

References

[1] O. Sharir, Y. Levine, N. Wies, G. Carleo, A. Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems (Center for Computational Quantum Physics 2020)